НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК _____

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 Інженерія програмного забезпечення

на тему: «Метод автоматизованого визначення образливого вмісту текстових повідомлень в соціальних мережах»

Виконала:
студентка VI курсу, групи КП-61м
Соколовська Анна Віталіївна _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Т.М. _____

Рецензент:
Доцент кафедри ММСА ІПСА, к.т.н., доцент,
Дідковська М.В. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2018 року

ЗМІСТ

СПИСОК СКОРОЧЕНЬ 4
ВСТУП 5
1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА СПОСОБІВ АВТОМАТИЗОВАНОЇ КЛАСИФІКАЦІЇ ПРИРОДНОМОВНИХ ТЕКСТОВИХ ДАНИХ 7
1.1 Огляд існуючих методів автоматизованої класифікації 7
1.2 Аналіз особливостей автоматизованої класифікації текстів з образливим змістом 15
1.3 Огляд особливостей класифікації образливого вмісту 18
1.4 Висновки 31
2 МЕТОД АВТОМАТИЗОВАНОГО ВИЗНАЧЕННЯ ОБРАЗЛИВОГО ВМІСТУ ТЕКСТОВИХ ПОВІДОМЛЕНЬ В СОЦІАЛЬНИХ МЕРЕЖАХ ..33
2.1 Аналіз специфіки визначення образливого вмісту текстових даних у соціальних мережах 33
2.2 Визначення особливостей стосунків між різними групами Інтернет-користувачів в соціальних мережах 39
2.3 Розроблений метод визначення образливого вмісту в повідомленнях Інтернет-користувачів в соціальних мережах 42
2.4 Висновки 47
3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗОВАНОГО ВИЗНАЧЕННЯ ОБРАЗЛИВОГО ВМІСТУ ТЕКСТОВИХ ПОВІДОМЛЕНЬ В СОЦІАЛЬНИХ МЕРЕЖАХ 49
3.1 Опис засобів розробки програмного забезпечення 49
3.2 Архітектура розробленого програмного забезпечення 55
3.3 Особливості реалізації програмного застосунку 60
3.4 Висновки 67
4 АНАЛІЗ ЕФЕКТИВНОСТІ МЕТОДУ АВТОМАТИЗОВАНОГО ВИЗНАЧЕННЯ ОБРАЗЛИВОГО ВМІСТУ ТЕКСТОВИХ ПОВІДОМЛЕНЬ В СОЦІАЛЬНИХ МЕРЕЖАХ 69
4.1 Використані метрики оцінки 69
4.2 Аналіз результатів 72
4.3 Висновки 76

5 ПОБУДОВА БІЗНЕС-МОДЕЛІ 77
5.1 Опис проблеми 77
5.2 Зацікавлені сторони 78
5.3 Комерційне рішення. Основні характеристики 81
5.4 Конкуренті переваги рішення 82
5.5 Клієнти. Сегменти ринку споживання 83
5.6 Унікальна ціннісна пропозиція 86
5.7 Доходи та витрати 87
5.8 Бізнес-модель 90
5.9 Висновки 92
ВИСНОВКИ 93
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 95
ДОДАТКИ 101

СПИСОК СКОРОЧЕНЬ

NLP – natural language processing.

SQL – structured query language.

POS – part-of-speech.

CBOW – continuous Bag of Words.

XML – eXtensible Markup Language.

LINQ – language Integrated Query.

FCL – framework Class Library.

CLR – common Language Runtime.

IDE – integrated Development Environment.

SOLID – single responsibility, open-closed, Liskov substitution, interface segregation, dependency inversion.

DI – dependency injection.

IoC – Inversion of control.

ООП – об’єктно-орієнтоване програмування.

ООД – об’єктно-орієнтовний дизайн.

ВСТУП

За останні роки використання мережі Інтернет значно зросло, збільшивши кількість постійних користувачів [1]. Можливості всесвітньої мережі широко використовуються у різних сферах діяльності людини, а особливої популярності за останні декілька років здобули соціальні мережі.

Сьогодні соціальні мережі широко використовуються для особистого спілкування, ведення блогів, реклами та навіть ведення бізнесу. Майже кожна компанія, від маленьких стартап-проектів та невеличких крафтових виробництв до величезних корпорацій та лідерів індустрії, має свою сторінку у соціальній мережі. Соціальні мережі стали місцем спілкування та об’єднання людей за інтересами. Проте там, де є велике скупчення людей та панує плюралізм думок, можливі виникнення палкої дискусії та поява образливого контенту.

Визначення образливого вмісту – це виявлення в тексті недопустимої мови, що спрямована на образу почуттів іншої людини або групи людей шляхом вживання нецензурних слів, цькування, приниження чужої гідності, дискримінацією за гендерною, етнічною, національною, расовою або іншою ознакою.

Зважаючи на той факт, що сучасні соціальні мережі – це не лише майданчик для спілкування, а й платформа для ведення бізнесу, поява образливого контенту, який жодним чином не модерується, може призвести до втрати рекламодавців та зниження прибутків самої соціальної мережі. Рекламодавці ж не бажають, щоб їх продукт асоціювався з негативними емоціями чи скандалами, які можуть вплинути на подальшу популярність товарів та прибутки. Саме тому соціальні мережі та рекламодавці зацікавленні в модерації даних повідомлень.

Такі гіганти ринку як Facebook, Twitter, YouTube та інші подібні платформи впроваджують відповідні політики безпеки, щоб захистити користувачів від різноманітних ризиків, наприклад, цькування,

дискримінації, проявів агресії [2]. В рамках дотримання вищезгаданих політик безпеки компаніями часто впроваджується модерація повідомлень, проте її якість здебільшого є низькою через те, що вона здійснюється за допомогою найманої робочої сили, а саме – людей-модераторів. Такий процес модерації є досить повільним та вимагає значних затрат на заробітну платню модераторів, соціальні виплати, страхові поліси (що в деяких країнах є обов'язковим), утримання робочих місць, оренду приміщень та інші супутні витрати. Також суттєвим недоліком при ручному визначенні образливого вмісту контенту є людський фактор, адже різні працівники можуть трактувати однаковий текст по-різному. В додаток до вищезазначених недоліків, в останні роки стрімке зростання чисельності Інтернет-аудиторії та відповідне збільшення кількості користувачів соціальних мереж, які продукують тисячі нових коментарів та постів щохвилинно, відчутно ускладнило задачу модерації коментарів та постів, призвело до збільшення фінансових та часових витрат на їх оброблення, а часом і взагалі унеможливило ручний аналіз їх вмісту.

Альтернативою даному підходу є забезпечення підтримки відповідними програмними продуктами автоматизованого виявлення образливого вмісту в текстових даних. Автоматизоване визначення образливого вмісту є відносно новим напрямком галузі обробки природної мови, саме тому, розроблення методів, алгоритмів, а також програмних засобів автоматизованого визначення образливого вмісту текстових повідомлень в соцмережах є актуальною задачею на сьогоднішній день.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА СПОСОБІВ АВТОМАТИЗОВАНОЇ КЛАСИФІКАЦІЇ ПРИРОДНОМОВНИХ ТЕКСТОВИХ ДАНИХ

1.1 Огляд існуючих методів автоматизованої класифікації

1.1.1 Формалізація задачі класифікації

Класифікація – один з розділів машинного навчання, присвячений вирішенню наступного завдання. Задача класифікації – формалізована задача, в якій є безліч об’єктів (ситуацій), розділених деяким чином на класи. Визначена кінцева безліч об’єктів, для яких відомо, до яких класів вони належать. Це безліч називається навчальною вибіркою. Класова приналежність інших об’єктів не відома. Потрібно побудувати алгоритм, здатний класифікувати довільний об’єкт з початкової множини.

Класифікувати об’єкт – значить, вказати номер (або найменування класу), до якого належить даний об’єкт. Класифікація об’єкта – номер або найменування класу, що видається алгоритмом класифікації в результаті його застосування до даного конкретного об’єкту.

Класифікація документів – одне із завдань інформаційного пошуку, яка полягає у віднесенні документа до однієї з декількох категорій на підставі змісту документа. Класифікація текстових даних або текстових документів – це відповідно віднесення письмового, друкованого документу або документу в електронному форматі, який представляє або передає певні дані у формі статті, листа, звіту тощо.

Класифікація може здійснюватися повністю вручну, або автоматично за допомогою створеного вручну набору правил, або автоматично із застосуванням методів машинного навчання.

Незважаючи на те, що дана дослідницька робота спрямована на реалізацію та застосування лише автоматизованих підходів до класифікації текстових даних (або текстових документів) за ознакою наявності в них

образливого вмісту, варто зазначити, що ручна класифікація також широко та активно використовується, а тому її короткий огляд слід розглянути слід в контексті даного дослідження, зважаючи на використання у сфері визначення образливого вмісту. Отже, класифікація не завжди здійснюється за допомогою комп'ютера. В більшості випадків задача класифікації текстових документів залишається мануальною задачею. Наприклад, у звичайній бібліотеці класифікація текстових документів – книжок на відповідні класи здійснюється книгам вручну бібліотекарем шляхом віднесення книжки до певної тематичної рубрики. Недоліками даного підходу є те, що подібна ручна класифікація є дорогою, адже вимагає оплати роботи найманого персоналу та відповідних супутніх витрат. Крім того, ручна класифікація є неприданою у випадках, коли необхідно класифікувати велику кількість документів, особливо з високою швидкістю. Проте, незважаючи на даний недоліки, визначення образливого вмісту в соціальних мережах базується в основному саме на мануальній класифікації. В переважній більшості існуючі механізми виявлення образливого вмісту в соціальних мережах здійснюються наступним чином: будь-який користувач може поскаржитися на іншого і за рішенням адміністрації сайту дане публічне повідомлення буде видаленим, а користувача, що його написав, може бути заблоковано. Даний підхід має безліч недоліків: починаючи від фінансової та часової затратності на здійснення класифікації і закінчуючи необ'єктивністю оцінки. Наприклад, різні люди мають різне сприйняття і можуть трактувати різноманітні рекомендації та гайдлайни по різному: один адміністратор може віднести повідомлення до образливого вмісту, інший – не буде вважати дане повідомлення образливим. Крім того, не варто відмітати людський фактор: фактор особистих симпатій та градації рівня сприйняття образливості. Саме тому є доцільним розглядати методи автоматизованої класифікації для вирішення задачі виявлення образливого вмісту в соціальних мережах.

1.1.2 Підходи до автоматизованої класифікації

Визначення образливого вмісту зазвичай визначають як одну з задач комп'ютерної лінгвістики, тобто мається на увазі, що ми можемо знайти і класифікувати текстові дані (далі текст) як образливий чи ні, використовуючи інструменти обробки природної мови (такі як тегер, парсери і ін.). Зробивши велике узагальнення, можна розділити існуючі підходи та виділити декілька основних підходів до автоматичної класифікації: *підходи, засновані на правилах, підходи, засновані на словниках, машинне навчання (з учителем та без учителя)*.

Перший підхід – автоматична класифікації природомовних текстових даних, що базується **на правилах** – полягає в написанні правил, за якими можна віднести текст до тієї чи іншої категорії (під категорією мається на увазі клас згідно термінології задачі класифікації). Наприклад, якби задача класифікації полягала у визначення до якого з предметів шкільної програми належить текст, то правила можна було б сформулювати наступним чином: «якщо текст містить слова похідна і рівняння, то віднести його до категорії математика», «якщо текст містить слова хлорофіл та фотосинтез, то віднести його до категорії біологія», «якщо текст містить слова електродинаміка та механіка, то віднести його до категорії фізика» і тому подібне. Для того, щоб сформулювати ряд необхідних правил згідно до сфери застосування, які потім будуть автоматично застосовуються до вхідних документів для їх класифікації, необхідний спеціаліст – людина, яка знайома з предметною областю та володіє навиками написання регулярних виразів. Регулярний вираз – це термін, що походить з англійської мови від словосполучення regular expressions, та тлумачиться як формалізована мова пошуку та здійснення маніпуляцій над підрядками в тексті, що базується на використанні метасимволів або ж іншими словами – символів підстановки (походить від англійського терміну wildcard characters). Символи підстановки – це

символи, що використовуються для заміни інших символів або послідовностей символів, що забезпечує приведення символів до очікуваних символьних шаблонів. Для пошуку за допомогою регулярних виразів використовується рядок-зразок, іншими словами шаблон або паттерн, що складається з символів та метасимволів, що задають правило пошуку. Для маніпуляцій над текстом додатково створюється рядок заміни, який може складатися з спеціальних службових символів. В декларативній мові програмування SQL (мова структурованих запитів для взаємодії користувача з реляційними базами даних), оператор LIKE є яскравим прикладом написання правил за допомогою регулярних виразів. Паттерн LIKE може виконувати різні запити за допомогою службових символів: LIKE 'a%' – знаходить будь-які значення, що починаються з символу "a"; LIKE '% a' – знаходить будь-яке значення, яке закінчується символом "a"; LIKE '%or%' – знаходить будь-які значення, які мають "or" у будь-якій позиції; LIKE '_r%' – знаходить будь-які значення, які мають "r" у другій позиції; LIKE 'a _% _%' – знаходить будь-які значення, що починаються з "a" і мають принаймні 3 символи у довжину; LIKE 'a% o' – знаходить будь-які значення, починаючи з "a" і закінчуючи "o". Даний приклад дозволяє сформувати спрощене уявлення про роботу автоматичної класифікації, заснованої на правилах.

Цей підхід кращий за ручну класифікацію, оскільки процес класифікації автоматизується і, отже, кількість оброблюваних документів є практично необмеженою. Більш того, побудова правил вручну може дати кращу точність класифікації, ніж при машинному навчанні. Однак для хорошої роботи системи необхідно скласти велику кількість правил, а створення і підтримання правил в актуальному стані (наприклад, найчастіше правила прив'язані до певного домену (наприклад, «ресторанна тематика») і при зміні домену («огляд фотоапаратів») потрібно заново складати правила. Наприклад, якщо для класифікації новин використовується ім’я чинного президента країни, відповідне правило

потрібно час від часу змінювати. Саме тому, автоматична класифікації, заснованої на правилах, вимагає постійних зусиль фахівця, підтримки актуальності правил та їх постійного рефакторингу. Багато комерційних системи використовують даний підхід, незважаючи на наведені недоліки, так як цей підхід є найбільш точним при наявності хорошої бази правил.

Отже, перевагами даного підходу є:

- висока точність;
- активне використання в комерційних системах.

Недоліками:

- трудоємний у створенні правил;
- точність зумовлюється написанням великої кількості продуманих правил;
- потребує багато затрат на підтримку.

Варто зазначити, що даний підхід не можна застосувати до будь-якої сфери діяльності. Найкращий баланс між перевагами та недоліками автоматична класифікація, заснована на правилах може показати у сферах людської діяльності, де текстові документи є підпорядкованими певним законам та алгоритмам побудови речень, наприклад, доцільно використовувати даний підхід до текстів з галузі юриспруденції, законотворення і тому подібне. Даний підхід не доцільно використовувати для визначення образливого вмісту, так як найчастіше визначення образливого вмісту використовується у сфері людської комунікації, спілкування, а відповідно у соціальних мереж. Користувачі мережі Інтернет є дуже винахідливими, тому майже щодня з’являються нові терміни, інфоприводи і тому подібне, що в свою чергу вимагає майже щоденного поповнення відповідних наборів правил. Відповідно до вищесказаного, затрати на написання правил будуть перевищувати переваги від точності (якої ще необхідно досягнути, написанням неймовірно громіздкої системи правил), а тому даний підхід до визначення

образливого вмісту недоцільно розглядати зважаючи на поставлену задачу.

Другий підхід, заснований на **словниках**, використовують так звані тональні словники або словники тональності (від англійських слів affective lexicons) для аналізу тексту. У простому вигляді тональний словник представляє з себе список слів зі значенням тональності (у контексті даного дослідження під тональністю мається на увазі ступінь образливості) для кожного слова. Щоб проаналізувати текст, можна скористатися наступним алгоритмом: спочатку кожному слову в тексті привласнити його значенням тональності зі словника (якщо воно присутнє в словнику), а потім обчислити загальну тональність всього тексту. Обчислювати загальну тональність можна різними способами. Найпростіший з них – обчислити середнє арифметичне всіх значень ступенів образливості слів наявних у досліджуваному текстовому документі. Більш складний підходом є навчання класифікатора (наприклад нейронної мережі) для визначення ступеню образливості вхідного документу.

Перевагою даного методу є:

- простота у застосуванні.

Недоліками:

- не універсальність;
- затратність часу на створення словника (якщо такого не існує).

Для задачі визначення образливого вмісту необхідно великий словник з різноманітними елементами специфічної лексики, що активно використовується в Інтернет. Наразі існують словники лише ненормативної лексики, проте в англійські мові використання нецензурного слова з різними допоміжними словами може надавати слову кардинально нового відтінку та змісту (інколи навіть гостро-позитивного [3]). Тому, як і в попередньому випадку затрати на створення та поповнення словника, та відносно інших способів невисоку точність (при

використання найпростішого способу «середнього арифметичного»), даний спосіб не є доцільним використовувати для вирішення даної задачі.

Третій загально відомий підхід до вирішення задач комп'ютерної лінгвістики ґрунтується на **машинному навчанні**. У цьому підході набір правил або, більш загально, критерій прийняття рішення текстового класифікатора, обчислюється автоматично з навчальних даних (іншими словами, проводиться навчання класифікатора). Навчальні дані – це деяка кількість хороших зразків документів з кожного класу, згідно яких проводиться класифікація. Для використання методів машинного навчання знадобиться розмічений набір текстів $\boldsymbol{T}$ для навчання системи, тобто кожному навчальному тексту присвоюється мітка певного класу. У машинному навчанні зберігається необхідність ручної розмітки (термін розмітка означає процес приписування документу певного класу з діапазону класів для класифікації). Але розмітка еталонних текстів – це значно простіше завдання за критеріями швидкості реалізації та необхідного рівня кваліфікації спеціаліста, ніж написання правил, як у першому підході до вирішення задач комп'ютерної лінгвістики. Для того, щоб сформувати набір еталонних текстів необхідно вибрати спосіб формалізації цих текстових даних, тобто деяким способом побудувати відображення $\boldsymbol{f}$ з безлічі текстів $\boldsymbol{T}$ в простір ознак $\boldsymbol{X}$

$$\boldsymbol{f: T \rightarrow X} \qquad (1)$$

Функцію $\boldsymbol{f}$ називають процедурою виділення ознак (від англійського терміну feature extraction). Така розмітка може бути проведена в звичайному режимі використання системи. Наприклад, в програмі електронної пошти може існувати можливість позначати листи як спам, тим самим формуючи навчальну множину для класифікатора – фільтра небажаної пошти. Або у соціальних мережах, модератори здійснюючи перевірку публічних повідомлень, на які надійшли скарги, теж можуть помічати такі текстові дані, як образливі, тим самим формуючи вибірку еталонних образливих текстів. Також, у модераторів частіше за все

існують критерії, за якими вони здійснюють видалення повідомлення, яке потрапляє під категорію повідомлень з образливим вмістом. Після визначення $\boldsymbol{f}$ та побудови простору ознак $\boldsymbol{X}$, кожного тексту з $\boldsymbol{T}$ поставлена у відповідність точка з $\boldsymbol{X}$. Після цього з'являється можливість застосовувати весь арсенал математичних методів для поділу всіх точок $\boldsymbol{X}$ на підмножини.

Таким чином, класифікація текстів, заснована на машинному навчанні, є прикладом **навчання з учителем**, де в ролі вчителя виступає людина, що задає набір класів і розмічає навчальну множину. Формально кажучи, в результаті необхідно побудувати відображення $\boldsymbol{g}$ з безлічі вектор-ознак $\boldsymbol{X}$ в безліч міток $\boldsymbol{L}$.

$$\boldsymbol{g : X \to L} \qquad (2)$$

Існують також інші способи постановки експерименту – **навчання без учителя**, але вони використовуються для вирішення іншого завдання – кластеризації або таксономії. У цих завданнях поділ об'єктів навчальної вибірки на класи не задається, і потрібно класифікувати об'єкти тільки на основі їх подібності між собою. Так завдання пошуку схожих текстів це є задача кластеризації точок з $\boldsymbol{X}$, завдання сортування текстів за темами зводитися до завдання класифікації точок з $\boldsymbol{X}$. У деяких прикладних областях, і навіть в самій математичній статистиці, через близькість завдань часто вже не розрізняють завдання кластеризації від завдань класифікації. Машинне навчання без вчителя є, напевно, найбільш цікавим і в той же час найменш точним методом аналізу образливого вмісту.

Перевагами машинного навчання без вчителя є:

- автоматичний;
- не потребує даних для навчання.

Недоліками:

- низька точність.

Деякі алгоритми для вирішення задач класифікації комбінують навчання з учителем з навчанням без учителя, наприклад, одна з версій

нейронних мереж Кохонена – мережі векторного квантування, яких навчають з учителем. Проте, для вирішення даної задачі очевидними є переваги навчання з учителем, а семе:

- автоматизований;
- велика кількість алгоритмів.

І єдиним недоліком для даного підходу до вирішення задачі пошуку образливого вмісту, що базується на машинному навчанні з учителем, є те, що даний підхід:

- вимагає даних для навчання;

Проте, створити навчальний набір – це простіше, ніж написання великої кількості правил чи створення громіздкого словника. А перевага у точності в порівнянні з навчанням без учителя перекреслює недолік у затраті часу на створення тренувального набору даних для класифікатора визначення образливого вмісту.

1.2 Аналіз особливостей автоматизованої класифікації текстів з образливим змістом

Визначення образливого вмісту частіше за все виявляється значно складнішою задачею, ніж цього можна було очікувати з першого погляду. На це є цілий ряд причин, що робить цю задачу комп'ютерної лінгвістики складною для автоматизації. Варто зазначити, що виявлення образливого вмісту є потенційно важким завданням для людей також, так як несе в собі велику кількість прихованих ускладнень.

Як зазначалося вище у попередньому розділі під час розгляду підходу до визначення образливого вмісту, заснованого на використанні тональних словників, однією з ключових переваг даного підходу визначалася простота реалізації та використання. Проте, однією з ключових особливостей природомовних текстових даних, що несуть у собі образливий вміст є навмисне заплутування слів і фраз, щоб уникнути ручної або автоматичної перевірки. Таке вуалювання образи в текстових

документах часто робить виявлення образливого вмісту досить важким завданням. Наприклад, образливі вислови можуть уникнути перевірки шляхом *заміни частини літер* з образливого слова *на службові символи*. Такі дії не дозволяють автоматизованим алгоритмам виявити образливий вміст, проте візуальне сприйняття написаного слова дозволить користувачам зрозуміти весь семантичний зміст. На кшталт, за допомогою такого підходу можна завуалювати слово «@$$hole» таким чином, що класифікатор його не розпізнає. Навіть один з найкращих та найбільших ресурсів для перекладу Translate від компанії Google перекладає дане слово як «@ $$ отвори», тобто сприймаючи «@» та «$$» як службові символи, якими вони і є. Аналогічно класифікатор для визначення образливого змісту відкинув би ці символи на етапі передобробки тексту, про який буде розказано детальніше далі, як службові і до самого етапу визначення образливого вмісту дійшло б слово «hole», яке саме по собі не є образливим. Проте для людського ока «@» та «$$» сприймаються як «a» та «ss» відповідно, що дозволяє уловити суть написаного.

Саме тому реалізувати успішне виявлення образи неможливою для простих метрик ключових слів, особливо тому що є багато варіантів для вихідного слова або фрази. З іншого боку, використання ключових слів з відповідних словників може привести до помилкових спрацьовувань. Можна зробити досить ефективний класифікатор з чорним списком (набором слів, що виражають образи або ненависть), однак, ці списки не є статичними і постійно змінюються. Таким чином, чорний список повинен регулярно оновлюватися, щоб йти в ногу зі зміною мови. Крім того, деякі образи, які можуть бути неприйнятні для однієї групи можуть бути абсолютно нормальними до іншої групи.

Також уникнути перевірки можна шляхом *відсутності пробілів у образливому реченні*. Таке речення буде важко сприйматися людським оком, проте при прикладенні достатньої кількості зусиль його зміст можна зрозуміти, шляхом вичленування окремих слів та фраз. Але на відміну від

людини, опрацювати такий текст буде неможливо за допомогою автоматизованих підходів. Наприклад, культова фраза з популярного американського фільму 1991 року «Кров та бетон», якби з неї видалили всі пробіли та знаки пунктуації виглядала б наступним чином:

«YoumotherfuckercomeonyoulittleassfuckwithmeehYoufuckinglittleassholedickheadcocksuckerYoufuckincomeoncomefuckwithmeIllgetyourassyoujerkOhyoufuckheadmotherfuckerFuckallyouandyourfamilyComeonyoucocksuckerslimebucketshitfaceturdballComeonyouscumsuckeryoufuckingwithmeComeonyouasshole».

Незважаючи на популярність даної цитати, відсутність в ній знаків пунктуації робить її схожу на вигляд на набір незрозумілих літер, призводить до помилкових спрацювань перекладача Google Translate та унеможливлює її пошук у системі Google. Саме тому визначення образливого вмісту у такому речення стає просто неможливим для існуючих автоматизованих підходів. Також існує проблема визначення образливого вмісту зважаючи на *контекст*. Так речення «Ця свиня товста.» та «Ти – товста свиня.» мають абсолютно різні забарвлення (перше – констатація факту, друге – відверта образа). Такі речення можна охарактеризувати терміном «ситуативна мова», який несе в собі наступне значення: окремі репліки можуть видатись не зрозумілими поза текстом. Також речення «Ми зобов'язані їх знищити!» теж не можна визначити як образливе не маючи контексту всього повідомлення (може йтися про якусь міжнаціональну ворожнечу, або ж про якийсь смертельний вірус, який закликають знищити). Справа в тому, що детекція образливого вмісту не обмежується тільки реченням. У деяких випадках доводиться брати інші речення до уваги, щоб вирішити, чи є текст образливим або носить випадки розпалювання ненависті.

Ще однією невід'ємною перепоною обробці природомовних текстів в цілому і визначення образливого вмісту зокрема є *сарказм*. Деякі користувачі публікують саркастичні коментарі в тому ж стилі, як люди, що

створюють образливий контент. Або навпаки пишуть все завуальовано і приємно, хоча мають на меті образу. Це дуже важко, як для людей так і для машин, отримати правильне розуміння написаного, оскільки це вимагає знання спільноти і, можливо, навіть самих учасників, які створюють даний контент.

Отже, існує ціла низка факторів, що ускладнює та погіршує якість визначення образливого вмісту в природомовних текстових даних. Звісно, врахувати всі ці фактори під час визначення образливого вмісту автоматизованими підходами просто неможливо, проте правильне застосування інструментів обробки природомовних текстів та врахування хоча б частини з цих особливостей в методах роботи програмних інструментів з оброблення природомовних текстів сприятиме підвищенню якості визначення останніми образливого вмісту повідомлень.

1.3 Огляд особливостей класифікації образливого вмісту

Процес створення системи аналізу образливого вмісту дуже схожий на процес створення інших систем, що вирішують задачі комп'ютерної лінгвістики, із застосуванням машинного навчання. По-перше, необхідно зібрати колекцію документів для навчання класифікатора. Потім кожен документ з навчальної колекції потрібно представити у вигляді вектора ознак та вказати «правильну відповідь» для кожного такого документу, тобто тип вмісту (наприклад, образливий чи нейтральний), за цими відповідями і буде навчатися класифікатор. По-друге, необхідно обрати алгоритму або метод автоматизованої класифікації та провести навчання обраного класифікатора на сформованих навчальних даних. Після отримання комплексу, що складається з навченого класифікатора та еталонних документів можна приступати до класифікації необхідних текстових документів.

Зазвичай в мережі вже можна знайти готові розмічені еталонні документи. Складності можуть виникнути при необхідності класифікувати

складні, нерозповсюджені серед широкого загалу типи текстів. В інших же випадків більшість навчальних даних можна знайти в мережі інтернет. Наприклад новинні тексти можна знайти на сайтах новинних ресурсах таких як видання Reuters, Bloomberg та інших. Також велика база готових даних для навчання класифікатора розміщена на сайті kaggle, який і позиціонує себе як спеціалізований сайт з data science та машинного навчання.

Кількість класів, на які ділять образливий вміст, зазвичай задається з специфікації системи. У більшості досліджень зазвичай розглядається задача бінарної класифікації, тобто виділяєтеся лише два класи, адже вона дає змогу досягти найвищої точності за найменших затрат для реалізації методу. Для галузі визначення образливого вмісту, також зазвичай виділяють лише два класи: «нейтральний» і «образливий». Класифікація образливого вмісту сильно перекликається з задачею визначення тональності [4]. Для визначення тональності класифікація на більш ніж два класи – це дуже складне завдання. Навіть з трьома класами дуже складно досягти хорошої точності незалежно від застосовуваного підходу. Аналогічно можна допустити, що додаткова класифікація образливого вмісту (наприклад, на «расизм», «нецензурну лайку», «національна нетерпимість») також буде складною у реалізації та досягненні прийнятної точності.

1.3.1 Способи передоброблення природомовних текстових даних

Також у процесі обробки природомовних текстових документів чільне місце займає передоброка тексту. На стадій передобробки вхідного тексту видаляються всі html теги, знаки пунктуації, службові символи. Дана операція може здійснюватися к самописними програмами так і за рахунок використання готових бібліотек (наприклад, за допомогою бібліотеки, реалізованою мовою python – «Beautiful Soup»). Наступним етапом передобробки є визначення в тексті так званих «стоп слів» – це

часті слова в мові, які в основному не несуть ніякого смислового навантаження (наприклад, в англійській мові це такі слова як «the, at, about ...»). Аналогічно попередньому етапу дані стоп слова можна виділяти самостійно або за допомогою вже існуючих реалізацій (наприклад, пакета Python Natural Language Toolkit (NLTK)). Після попередньої обробки вихідного тексту ми отримуємо наступне: [Biography, part, feature, film, remember, going, see, cinema, originally] – тобто набір слів. Даний етап можна узагальнити одним словом – *токенізація*. Токенізація – у лексичному аналізі це процес розділення потоку тексту на слова, фрази, символи або інших конструктивних елементів, так звані токени [5].

У складних природних мовах (таких як українська мова) одне і те ж слово може приймати різні форми (відмінки), і в словник частотного аналізу можуть потрапляти всі словоформи, що відрізняються префіксами і/або закінченнями. Через це може сильно збільшуватися розмір словника і відповідно розмір набору даних для навчання, що може викликати падіння продуктивності системи і погіршувати узагальнюючі здатності класифікатора (перенавчання).

Існує кілька способів вирішення цього завдання, які відносяться до одного етапу – *нормалізації*.

Перший підхід це *лематізація* [6] – всі слова в тексті приводяться до нормальної форми (нормальною формою слова є слово у називному відмінку однини). В основі даного підходу лежить словник, в якому вже записано велику кількість слів і їх форм, та в першу чергу слово перевіряється за словником. І якщо воно там є, то зрозуміло, в який початковій формі воно наводиться. Якщо слова немає, то за певним алгоритмом виводиться спосіб зміни даного слова, і на основі цього способу вже робляться висновки, які початкові форми можуть бути у цього слова. Цей підхід підходить для нових, невідомих слів, але при цьому, оскільки лематизація вимагає словників мов, на яких написаний текст, та

виконує довгий пошук для всіх слів тексту їх нормальної форми по словниках, то це може суттєво знижувати продуктивність.

Другий метод це *стемінг* [6] – виділення основи слів шляхом відкидання приставок і закінчень. Цей спосіб нормалізації тексту працює набагато швидше ніж лематізація, проте він є менш точним. Наприклад, даний підхід не завжди працює – наприклад, деякі слова, коли змінюється форма, змінюються цілком. Так у слова «є» його минулий та майбутній часові форми – це «був» і «буде». Зрозуміло, що скільки б букв з кінця ви не відрізали, ви не отримаєте з цих трьох слів одне і те ж в результаті стемінгу.

Також можна використовувати *хешування* – необхідно побудувати функцію відображає безліч слів в безліч хеш-кодів фіксованої довжини, причому, прагнути до того, щоб схожі слова (словоформи одного слова) мали однаковий хеш. Перед виконанням частотного аналізу, всі слова в тексті і словнику замінюються хешкодамі, що дозволяє скоротити кількість непотрібних повторів.

1.3.2 Обробка природомовних текстових даних

Наступним етапом обробки вхідного тексту є виділення суттєвих ознак. Суттєві ознаки зазвичай виділяють для формування конкретного лінгвістичних даних (текстових корпусів) для подальшої роботи класифікатора. У лінгвістиці корпус (множинний або текстовий корпус) – це великий і структурований, оброблена за певними правилами набір текстів, які використовуються в якості бази для дослідження мови (у конкретному випадку класифікації). Найчастіше лінгвістичні корпуси досліджують разом з частотними словниками (ще їх називають списками), що представляють собою набір слів певної мови або підмови разом з інформацією про частоту, з якою вони зустрічаються. Частотні словники забезпечують можливість порівняти два корпуси, щоб визначити слова, найбільш характерні для кожного з них.

Існує декілька підходів виділення суттєвих ознак [7]:

- Лінгвістичні риси:
    - Лексикон;
    - Лінгвістика.
- N-грами:
    - Токенні N-грами;
    - Символьні N-грами.
- Синтаксичні.
- Розподілені:
    - Bag-Of-Words;
    - Word2vec;
    - Comment2vec.

Перш ніж перейти до розгляду даних підходів необхідно звернути увагу на такий важливий термін, як шумний текстовий аналіз. Шумний текстовий аналіз – це процес вилучення інформації, мета якого – автоматично вичленовувати структуровану або напівструктуровану інформацію з шумних неструктурованих текстових даних. Шумний текстовий аналіз набуває все більшого значення, оскільки багато загальних програм створюють шумові текстові дані. Шумні неструктуровані текстові дані найчастіше генеруються у неформальних місцях спілкування, таких як онлайн-чат, текстові повідомлення, електронні листи, дошки оголошень, групи новин, блоґи та веб-сторінки. Також документи історичної тематики, що написані з вживанням специфічної мови також можуть вважатися шумними щодо сучасних знань про сучасну мову спілкування. Такий вид текстів містить важливі історичні, релігійні, інколи медичні терміни, що вже вийшли із вжитку в сучасній мові. Такі тексти наповнені історизмами та архаїзмами, що значно ускладнює їх обробку. Крім того, текст, створений шляхом обробки спонтанної мови за допомогою автоматичного розпізнавання мови та друкованого або рукописного тексту за допомогою оптичного розпізнавання символів, містить безліч шумів. Отже,

згенерований таким чином текст, зазвичай є дуже шумним, тобто містить орфографічні помилки, скорочення, нестандартні слова, повторення, відсутність знаків пунктуації, відсутність інформації про коментар до листа, слова, що заповнюють паузу, такі як «хм», «гм», «ммм» (у англійській мові «um» та «uh») та інші шуми. Такий текст можна побачити у великих кількостях у різноманітних чатах, текстах коротких повідомлень (СМС або англійською SMS), що немало важливо повідомлень користувачів соціальних мереж та інше. На відміну від історичних текстів, тут переважають неологізми, абревіатури, сленгові вислови та досить таки часто меми. Мем (англ. Meme) – одиниця культурної інформації. Мемом може вважатися будь-яка ідея, символ, манера або образ дії, свідомо чи несвідомо передаються від людини до людини за допомогою мови, листи, відео, ритуалів, жестів і т. д. З популяризацією мережі Інтернет меми отримали нове середовище для поширення і лягли в основу особливого соціального явища – інтернет-мемів. Інтернет-меми – це інформація (посилання, тексти, картинки, навіть розмовні конструкції), яка зазвичай передається користувачами один одному прямо через Інтернет мережу. Зазвичай це робиться з метою розваги, але аналогічним способом може поширюватися і інша інформація, в тому числі провокаційного характеру (в тому числі і образливий вміст).

Лінгвістичні риси

Лінгвістика – це наука, що вивчає мову в усій складності її прояву [8]; наука про мову взагалі й окремі мови світу як індивідуальних її представників [9]. У контексті машинного навчання лінгвістичний підхід розглядається у контексті зменшення кількості «шумів» в тексті. У генеративній (або породжувальній) лінгвістиці лексика або лексикон є повним набором всіх можливих слів на мові. У цьому сенсі дитина, діти, дитяча – три різних слова в українській лексиці. У системно-функціональному мовознавстві лексика або лексичний елемент – це спосіб, яким називається конкретна річ або тип явища. Оскільки лексика з

системно-функціональної перспективи є способом за допомогою якого можна назвати предмет або явище, вона не обмежується одним конкретним словом та може бути реалізована багатьма граматичними словами, наприклад словосполучення «Білий Дім», «Нью-Йорк Сіті» або «серцевий напад» складаються з декілька значущих слів, проте несуть одне смислове навантаження. Крім того, різні слова, такі як дитина, діти та дитячий хоча і звучать та пишуться по-різному, але можуть реалізувати (називати) одну і ту саму лексичну одиницю.

Функції, що базуються на лінгвістичних рисах, призначені для прямого пошуку лайливих слів (аналогічний підхід використовувався в раніше існуючих списках ненависті), але також елементів, що стосуються необразливої мови, наприклад, вживання слів ввічливості або модальних дієслів. Модальний дієслово – це тип дієслова, який використовується для позначення способу – тобто: правдоподібності, здібностей, дозволу та обов'язку [10].

У дослідженні 2016 року щодо визначення образливої мови в контенті Інтернет-користувачів [7] до функції, що базуються на лінгвістичних рис, також було включено наступні характеристики:

- довжина коментарю в токенах;
- середня довжина слова;
- кількість знаків пунктуації;
- кількість крапок, знаків питання, лапок та повторюваних знаків пунктуації;
- кількість однобуквених токенів;
- кількість великих літер;
- кількість URL-адрес;
- кількість токенів з небуквеними символами в середині;
- кількість дискурсних зв'язків, заснована на [11];
- кількість слів ввічливості;

- кількість модальних слів (для вимірювання хеджування та довіри доповідача);
- кількість невідомих слів у порівнянні зі словником англійських слів (мається на увазі для вимірювання унікальності та будь-яких орфографічних помилок);
- кількість образ (образливих висловів) і слів ненависті з чорного списку.

Варто зазначити, що під дискурсивними зв'язками мається на увазі, слово або фраза, яка відіграє певну роль у керуванні потоком і структурою дискурсу. Оскільки їх основна функція знаходиться на рівні дискурсу (послідовності висловлювань), а не на рівні висловлювань чи речення, маркери дискурсу відносно синтаксичні, і зазвичай не змінюють умовного значення речення [12].

За результатами дослідження [7], яке проводилося на наборі даних для коментарів вилучених з двох доменів – Yahoo! Finance та Yahoo! News (коментарі з з фінансів та новин) у співвідношенні 80% коментарів для навчання та 20% для тестування, показали результати у 0.539 та 0.522 точності для фінансів та новин відповідно.

За словами авторів, всі коментарі були надані модераторами – співробітниками компанії Yahoo, основною функцією яких були надання редакційних міток для різних коментарів. Всі працівники мали принаймні ступінь бакалавра та були знайомі з концепцією судження текстових уривків для різних типів завдань та вимог до коментаря. Перш, ніж взяти на себе фактичне завдання модерації, вони пройшли навчання, щоб ознайомитись з настановами щодо текстових рішень.

N-grams

N-грам – це послідовність з n елементів [13]. Елементи ж, з яких складається послідовність N-грама в свою чергу з семантичної точки зору може бути послідовність звуків, букв, складів або слів, у рідких випадках словосполучень (частіше за все це є сталі вирази). Найбільш популярними

(за критерієм їх використання у різноманітних дослідженнях). Є N-грами, послідовність елементів яких представляє з себе послідовність слів, та стійких словосполучень. Стійкі або фразеологічні словосполучення характеризуються терміном колокація. Колокація – це особливий вид словосполучення, що має ознаки синтаксично і семантично цілісної одиниці, в якому вибір одного з компонентів здійснюється за змістом, а вибір другого залежить від вибору першого. До колокація також зазвичай зараховують складові топоніми, антропоніми та інші часто спільно вживаються іменування. N-грам характеризуються своєю розмірність, тому N-грам розміру з одного елемента називається «юніграм», з двох послідовних елементів має назву «біграм», послідовність з трьох елементів – «триграм». Послідовності N-грам з чотирьох і вище елементів називаються 4-грамам, 5-грамам і так далі, тобто N замінюється на значення кількісті елементів у послідовності.

N-грами широко використовуються різних областях наук, зокрема в обробці природної мови, де N-грами використовується в більшості випадків для передбачення на основі імовірнісних моделей. N-грамна модель розраховує ймовірність останнього слова N-грами, якщо відомі всі попередні. При використанні цього підходу для моделювання мови передбачається, що поява кожного слова залежить тільки від попередніх слів [14].

Також часто N-грам використовують для задач категоризації тексту та мови, в задачах пошуку плагіату в текстових даних, так як розділені на фрагменти тексти у представленні N-грамам легко порівняти між собою та визначити ступінь подібності досліджуваних тестів. N-грами застосовуються для перевірки правопису та виправлення помилок, а також для створення функцій, які дозволяють отримувати знання з текстових даних (наприклад, «зв’язувати» між собою слова, що мають тенденції застосовуватися парами).

За результатами дослідження [7], де було використано символи n грам (від 3 до 5 символів, пробіли включені) і токени юніграмів та біграмів для моделювання типів свідомого або несвідомого забарвлення образливих слів, для коментарів під фінансовими та новинними статтями було отримано точність 0.726 та 0.769 відповідно.

Syntactic

Синтаксис – це розділ лінгвістики, завданням якого є вивчення будови та функціональної взаємодії різних частин мови в реченнях та словосполученнях. Синтаксис словосполучень – це підрозділ синтаксису, що встановлює синтаксичні властивості окремих слів як частин мови, мається на увазі правила їх сполучуваності з іншими словами. Синтаксис речень –це підрозділ синтаксису, що в свою чергу спрямований на дослідження типів речень, їх ознак, зв'язків слів, словосполучень та інших сполук у складі речень. Використання розбору (парсингу) природної мови є поширеним для завдань, починаючи від аналізу настроїв [15] до прогнозування найкращої відповіді в аналізі питань-відповідей (Question Answering) [16].

Для синтаксичного аналізатору можна отримати функції з різних сторонніх бібліотек, наприклад з аналізатора залежностей ClearNLP v2.0. Функції, по суті, різні типи кортежів, що використовують слова, POS-теги та відносини залежностей. POS-тегування (від англійського POS tagging, part-of-speech tagging, що дослівно перекладається як розмітка частин мови або іншими словами автоматична морфологічна розмітка) – це етап автоматичної обробки тексту, завданням якого є визначення частини мови і граматичних характеристик слів в тексті (корпусі) з присвоєнням їм відповідних тегів.

Отже, до функцій синтаксичного аналізу відносяться, що визначають:

- батька вузла;
- прабатька вузла;

- POS-теги батька;
- POS-теги прабатька;
- кортеж, що складається з слова, батька та прабатька;
- дітей вузла;
- кортежі, що складаються з перестановок слова або його POS-тегів, мітки залежностей, що зв’язують слово з його батьком, а також батьків або його POS-тегів.

Використання цих функцій полягає у захопленні більш широких залежностей між словами, ніж це можуть зробити n-грами. Наприклад, у реченні: студенти є свинями нижчого класу, модель, побудована за допомогою виділення n-грамів не зможе зв’язати між собою слова «студенти» та «свині», однак, використовуючи аналізатор синтаксичних залежностей, можна згенерувати наступні кортежі: «є–студенти–свині», де іменники «студенти» та «свині» – це діти дієслова «є».

Проте, у дослідженні [7], де було використано аналізатор залежностей ClearNLP v2.0, для коментарів під фінансовими та новинними статтями було отримано точність 0.689 та 0.748 відповідно, яка поступається результатам, що отримані за допомогою моделі, побудованої на n-грамах (0.726 та 0.769).

Distributional Semantics Features

Розподільча семантика – це дослідницька область лінгвістики, яка розробляє та вивчає теорії та методи кількісної оцінки та класифікації семантичних подібностей (ступеня семантичної близькості) між лінгвістичними елементами на основі їх дистрибутивних властивостей у великих зразках мовних даних (текстових корпусах). Основну ідею розподільчої семантики можна підсумовувати в так званій гіпотезі розподілу: лінгвістичні елементи з подібними розподілами мають аналогічні значення. Розподіл забезпечується шляхом формування для кожного слова мови свого контекстний вектор. Безліч векторів формує словесний векторний простір.

Використання розподільчої семантики набуло широкої популярності в реалізації різноманітних програмних засобів з обробки природної мови.

Дослідження в більшості своїй пов’язані з представлень мови та тексту [17] [18] [19], які покращують попередні спроби моделювати лексичну семантику за допомогою векторних просторових моделей [17].

Так, як виявлення образливого вмісту є відносно новий завданням комп’ютерної лінгвістики, тому на сьогоднішній день існує небагато досліджень, що пов’язані з виявленням образливого вмісту за допомогою використання розподілених рис. Даний підхід був застосований застосований для виявлення образливого вмісту мовою у дослідженнях «Hate speech detection with comment embeddings» [20] та «Abusive Language Detection in Online User Content» [7].

Найбільш популярними підходами до визначення розподільчих рис є bag-of-words представлення (мішок слів) та Word2vec (слово до вектору).

Представлення bag-of-words (або «мішка-слів») є спрощеним представленням взаємозв’язків слів у корпусі, що використовується при обробці природних мов та пошуку інформації. У даному представленні текст (наприклад, речення або документ) представляється як сумка (від англійського слова bag) або мультисет слів даного тексту, ігноруючи граматику і навіть порядок слів, але зберігаючи множинність.

Word2vec – представлення, яке базується на дистрибутивній семантиці та векторній поданні слів. Word2vec в якості свого вводу приймає великий корпус тексту і створює векторний простір, при цьому кожне унікальне слово у корпусі призначається відповідним вектором у просторі. Вектори слів розташовуються у векторному просторі таким чином, щоб слова, які поділяють загальні контексти в корпусі, знаходяться в безпосередній близькості один від одного в просторі [17]. Тобто, векторне подання ґрунтується на контекстної близькості: слова, що зустрічаються в тексті поруч з однаковими словами (а отже, мають схожий зміст), у векторному поданні матимуть близькі координати векторів-слів.

Отримані вектори-слова можуть бути використані для обробки природної мови та машинного навчання.

У дослідженні «Abusive Language Detection in Online User Content» також використовувалось представлення comment2vec (коментар до вектору), яке забезпечувало відображення кожного коментаря в унікальному векторі в матриці, що представляє коментарі, і кожне слово зв'язувало з унікальним вектором в матриці, що представляє слова. Це дозволило об'єднати вектор коментарів та вектори слів, щоб передбачити наступне слово в контексті, точніше, ймовірність розподілення образливого слова залежить не тільки від фіксованого числа навколишніх слів, але також залежить від конкретного коментарю. Таким чином, кожен коментар було представлено вигляді низько розмірних щільних векторів, який навчається прогнозувати слова в коментарях і долає слабкі місця векторного представлення слова.

За результатами дослідження [7], для коментарів під фінансовими та новинними статтями було отримано точність 0.653 для фінансових коментарів та 0.698 для новинних коментарів за допомогою word2vec та comment2vec дав точність 0.680 та 0.758 відповідно.

У даному дослідженні [7] мішок слів не розглядався, але, CBOW (Continuous Bag of Words) або «безперервний мішок зі словами» лежить в основі одного з двух основних алгоритму навчання word2vec (порядок слів контексту не впливає на результат ні в одному з цих алгоритмів), і дозволяє, використовуючи поточний слово, передбачати навколишні його слова. Цей факт дозволяє зробити припущення, що результати роботи мішка слів є близькими до word2vec, але значно простішим в реалізації.

Після визначення суттєвих ознак за допомогою будь-якого з запропонованих підходів безпосередньо виконується класифікація за допомогою машинного навчання. Наразі існує безліч методів машинного навчання з вчителем, а найбільш популярними: наївний баєсівський

класифікатор, метод опорних векторів, метод k-найближчих сусідів, випадковий ліс та інші.

1.4 Висновки

Аналіз існуючих підходів до автоматизованої обробки природомовних текстових даних показав, що для визначення образливого вмісту природномовних текстів найбільш підходять підходи, засновані на машинному навчанні з вчителем. Розглянуто конкретні прикладні реалізації даного підходу, визначено їх переваги та недоліки.

Розглянуто особливості визначення образливого вмісту. Визначено низку параметрів, які можуть вплинути на результати обробки текстів. Розповсюдженими способам уникнення детекції образливого вмісту є заміна частини літер на службові символи та відсутність пробілів у образливому реченні. Також явищами, що суттєво ускладнюють, а інколи навіть унеможливлюють визначення образливого вмісту є складність врахування контексту та визначення сарказму.

Також було розглянуто етапи передобробки природомовних текстів, визначено переваги та недоліки кожного з підходів. Проведено аналіз попередніх досліджень. Як вже зазначалося вище, автоматизоване визначення образливого вмісту є відносно новим напрямком галузі обробки природної мови, саме тому не всі з підходи, що застосовуються до задач комп’ютерної лінгвістики були досліджені у контексті задачі визначення образливого вмісту. Більшість розробок в даному напрямку поки здебільшого залишаються в рамках науково-дослідних робіт та не враховують особливостей спілкування у соціальних мережах, що в перспективі могло б покращити якість детекції образливого вмісту повідомлень Інтернет-користувачів при їх модерації.

Отже, актуальною задачею є розроблення методів, а на їх основі програмного забезпечення для автоматизованого визначення образливого

вмісту текстових повідомлень Інтернет-користувачів, враховуючи особливості спілкування в соціальних мережах.

Таким чином, з огляду на вищезазначене, метою даної роботи стало підвищення точності визначення образливого вмісту текстових повідомлень в соціальних мережах шляхом розроблення нового методу визначення образливого вмісту в природномовних текстових даних з урахуванням специфіки спілкування в соцмережах.

2 МЕТОД АВТОМАТИЗОВАНОГО ВИЗНАЧЕННЯ ОБРАЗЛИВОГО ВМІСТУ ТЕКСТОВИХ ПОВІДОМЛЕНЬ В СОЦІАЛЬНИХ МЕРЕЖАХ

2.1 Аналіз специфіки визначення образливого вмісту текстових даних у соціальних мережах

Соціальна мережа – це структура, утворена індивідами або організаціями, яка відображає різноманітні зв’язки між ними через різні соціальні взаємовідносини, починаючи з випадкових знайомств і закінчуючи тісними родинними зв'язками. Починаючи з 2001 року стали з’являтися сайти, в яких використовувалась технологія під назвою Коло друзів. Ця форма соціальних мереж набула широкої популярності, яка постійно зростала і сьогодні широко використовується у віртуальних спільнотах (наразі, існує більш ніж 200 сайтів з можливостями організації соціальних мереж). У 2004 році була створена найбільша на сьогоднішній день соціальна мережа у світі Facebook [21].

В більшості випадків відвідувачі соціальних мереж обмінюються особистими повідомленнями, пишуть пости на своїх приватних чи публічних сторінках, коментують щось на сторінках інших користувачів чи різноманітних спільнот. Перший пункт з наведеного вище переліку в контексті виявлення образливого вмісту розглядати не доцільно, адже приватні повідомлення не повинні перевірятися, бо це порушує конфіденційність розмови та приватність співрозмовників. У всіх інших випадках модерація коментарів та постів є доцільною.

Слід відмітити, що існуючі на сьогоднішній день підходи до модерації повідомлень та відповідні методи є недостатньо ефективними за критеріями точності та швидкості оброблення даних, а також вимагають значних фінансових затрат. Розглянемо їх більш детально.

По-перше, на разі активно використовується ручна модерація. Недоліками даного підходу є: низька швидкість оброблення даних, висока

вартість даного процесу та незадовільна якість отримуваних результатів (різні люди можуть по-різному трактувати як образливий вміст так і правила, за якими повинна здійснюватися модерація). Ще одним з підходів так званої «ручної модерації» є система скарг інших користувачів. На сьогоднішній день кожен користувач може поскаржитися на пост або коментар іншого користувача. Якщо таких скарг буде декілька, то даний пост чи коментар буде автоматично заблоковано (і немає жодного значення, де було написано даний коментар, для якого кола осіб він був відкритим і чи взагалі містив образливий вміст). Часто скарги є недобросовісними: користувач може поскаржитися на будь-що, навіть на цілком нейтральний та необразливий пост чи коментар. Дана опція несе в собі один відчутний недолік – так звані замовні атаки, коли людям платять за скарги на чиїсь пости, наприклад політиків. Крім того, дані атаки можуть спричинятися не реальними людьми, а «фабриками ботів» (спеціально написаними програмами, що емулюють діяльність справжніх користувачів в соціальних мережах). Цією слабкістю даного підходу активно користуються зловмисники, наприклад, здійснюючи замовні атаки на сторінки відомих людей.

По-друге, на сьогоднішній день існують методи та способи автоматизованого визначення образливого вмісту текстових даних. Недоліками даних розробок є те, що вони дають дуже загальну оцінку повідомленню щодо наявності в ньому образливого вмісту (текст образливий чи нейтральний) та не беруть до уваги специфіку спілкування у соціальних мережах. Якщо не враховувати дану специфіку, соціальні мережі з місця вільного висловлення думок можуть перетворитися на місця з обмеженням свободи слова та жорсткою цензурою, а відповідно і втратять прихильність більшості аудиторії. Саме тому автоматизоване визначення образливого вмісту в повідомленнях Інтернет-користувачів є особливо актуальним.

Таким чином, на основі результатів проведеного дослідження існуючих підходів до модерації текстових повідомлень в соціальних мережах, а також відповідних методів та способів автоматизованого визначення образливого вмісту такого роду контенту можна зробити висновок, що підвищення ефективності процесу детекції образи у тому чи іншому повідомленні Інтернет-користувачів повинне базуватися, в першу чергу, на детальному вивченні специфіки публікацій Інтернет-користувачів в соціальних мережах та формулюванні переліку їх характеристик, які можуть бути враховані під час автоматизованого визначення наявності образливого вмісту, а також на обґрунтованому визначенні місця та способу врахування характеристик повідомлень у соцмережах, а також формату стосунків між різними типами та групами відвідувачів соціальних мереж у загальному процесі детекції образливого контенту.

Текстові повідомлення Інтернет-користувачів в соціальних мережах мають певні особливості, що відрізняють їх серед інших видів природномовних текстових даних та ускладнюють задачу їх автоматизованого оброблення та аналізу в цілому і визначення образливого вмісту зокрема. Розглянемо їх більш детально.

Спілкування у соціальних мережах найчастіше можна віднести до розмовного стилю мовлення, в якому переважають ознаки побутового спілкування. Саме тому особливості побутового спілкування є релевантними і для спілкування в соціальних мережах, наприклад [22]:

1. Мова найчастіше спонтанна, непідготовлена, відображає хвилинний настрій, відчуття, пориви. У ній виражена індивідуальність кожного суб’єкта.
2. Переважає діалог, хоча наявний полілог. Окремі репліки можуть видатись не зрозумілими поза текстом (характеризується терміном «ситуативна мова»).
3. Емоційний рівень спілкування надзвичайно високий.

4. Наявні умовчання, натяки, алюзії, посилання на події, відомі тільки співбесідникам.
5. У широкому вжитку сленгові слова та скорочення.

Зупинимось на декількох характерних рисах повідомлень Інтернет-користувачів, що утруднюють виявлення в них образливого вмісту.

По-перше, варто зазначити, що мова спілкування в соціальних мережах характеризується використанням постійно змінюваного *сленгу* та в цілому не обмежена жорсткими рамками, завдяки чому постійно поповнюється неологізмами та абревіатурами. Також характерним є використання нецензурних слів, які у комбінації з різними допоміжними словами деколи можуть надавати словосполученню чи фразі кардинально нового відтінку та змісту (інколи навіть гостро-позитивного [3]). Наприклад, комбінація слова «fuck» з іншими словами, таким як «yeah» або «awesome» допомагає підкреслити захоплення оратора суб'єктом обговорення, а додавання форми слова «fuck» між ім'ям і прізвищем особи дозволяє підкреслити, що ви говорите про прекрасну людину – «Oprah Fucking Winfrey». Крім того, деякі образи, які є неприпустимими для однієї групи людей, можуть сприйматися не як образа іншою групою.

По-друге, інколи використання тих чи інших слів та фраз може суперечити політикам безпеки соціальних мереж, тому для уникнення блокування недобросовісні користувачі соціальних мереж також вдаються до *навмисного заплутування слів і фраз*. Як у сфері визначення образливого вмісту, так і в соціальних мережах зокрема залишаються проблеми Такі вирази як «ni99er» чи відсутність пробілів в образливому реченні ускладнюють успішне виявлення образи, а інколи робить дану задачу взагалі неможливою.

Ще однією проблемою при визначенні образливого вмісту в повідомленнях Інтернет-користувачів є *врахування контексту*, тобто де та чи інша потенційно образлива фраза була сказана. До так званої ситуативної мови можуть бути віднесені речення «I hate fucking

cockroaches.» та «My neighbour is fucking cockroach.», які мають абсолютно різне забарвлення (перше виражає ненависть до тарганів і навряд чи може задіти чужі почуття, друге – відверта образа конкретної людини) проте майже не відрізняються лексично, проте абсолютно різні семантично. Також у другому повідомлення може в соціальній мережі бути прикріплене посилання на сторінку іншого користувача, якого обізвали тарганом. Також речення «You are fucking cockroach!» теж не можна визначити як образливе, не маючи на руках контексту всього повідомлення (може йтися про образу певної людини, або це може бути референс до культової фрази персонажа Аль Пачіно – Тоні Монтни з фільму 1983 року «Обличчя зі шрамом» у пості, де описуються користувачем враження від фільму). З точки зору специфіки повідомлень в соціальних мережах контекст може не обмежуватися лише одним повідомленням: це може бути низка повідомлень, що об'єднані спільною ідеєю, і лише при розгляді всього набору таких повідомлень можна визначити образливий вміст, позаяк окремо один від одного повідомлення можуть нести нейтральне забарвлення.

Ще однією особливістю спілкування у соціальних мережах та одночасно суттєвою перепоною при обробленні природномовних текстів в цілому і визначенні образливого вмісту зокрема є *сарказм*. Такі жорсткі насмішки особливо полюбилися користувачам мережі Інтернет, зокрема соціальних мереж. Складність як для людей так і для машин, отримати правильне розуміння написаного, полягає у тому, що саркастичне повідомлення може відкриватися позитивним судженням, але в цілому завжди містить негативне забарвлення і вказує на недолік людини, предмета або явища, тобто того, щодо чого відбувається.

Наведений перелік рис, характерних для спілкування Інтернет-користувачів у соціальних мережах, був би неповним, якби ми не згадали про емодзі – новий спосіб висловлення думок та надання повідомленням емоційного забарвлення. Емодзі («емодзі» походить з японської мови від

слів «картинка» та «знак», «символ») – особлива мова ідеограм і смайлів, що широко використовують в електронних повідомленнях та на сторінках сайтів, яка отримала неймовірну популярність в останні роки та стала новою мовою для комп'ютерних комунікацій (Computer-Mediated Communication або CMC). Враховуючи те, що користувачі генерують великий обсяг емодзі, нещодавно були проведені дослідження для того, щоб зрозуміти спосіб використання емодзі через додатки [23], на різних платформах [24] та навіть у різних культурах [25]. У порівнянні з традиційними інформаційними представленнями, такими як текстові повідомлення, картинки або навіть смайлики, емодзі вважаються більш жвавими, виразнішими та більш семантично насиченими, а саме тому цінуються користувачами Інтернету. Наприклад, повідомлення I love you може бути інтерпретованим у емоджі-речення наступним чином: I 💕 👉 👤.

На сьогоднішній день емодзі жодним чином не враховуються при аналізі природномовних текстових даних. Але зважаючи на популярність даного способу обміну повідомленнями в мережі Інтернет в цілому, та соціальних мережах зокрема, нехтувати емодзі при класифікації текстів є неправильним.

Як вже зазначалося вище існуючі механізми упередження появи образливого вмісту в соціальних мережах передбачають наступне: кожен користувач може поскаржитися на пост або коментар іншого користувача. Такий підхід породжує тенденції, що призводять до зменшення свободи слова в мережі Інтернет. Адже, навіть якщо людина висловлює образливу для когось думку на своїй закритій персональній сторінці, то користувач, якого ображає даний контент, може просто відписатися або додати до «чорного списку» користувача, що його розповсюджує.

Інша справа – публічні сторінки, на які можуть зайти діти або дуже вразливі люди. Ступінь відповідальності для даних груп користувачів має бути іншим.

Також важливо наголосити на тому, що зазвичай при визначенні образливого вмісту повідомлень користувачів соцмереж опрацьовують лише сам текст повідомлення, проте не враховують його контексту на рівні зв'язків користувачів в соціальних мережах [26]. Врахування зв'язків між користувачами в соціальних мережах дозволить зменшити кількість необґрунтованих блокувань повідомлень на приватних сторінках, тим самим запобігаючи жорсткому цензуруванню повідомлень в соціальних мережах.

Звісно, взяти до уваги всі вищезгадані особливості повідомлень Інтернет-користувачів під час визначення в них образливого вмісту за допомогою автоматизованого інструментарію просто неможливо (наприклад, сарказм), проте врахування частини з наведених вище ознак може підвищити якість визначення наявності образливого вмісту.

2.2 Визначення особливостей стосунків між різними групами Інтернет-користувачів в соціальних мережах

Як вже зазначалося вище соціальна мережа відображає розмаїті зв'язки між різноманітними індивідами через соціальні взаємовідносини, починаючи з випадкових знайомств і закінчуючи тісними родинними зв'язками. Умовно соціальні мережі можна поділити за наступними ознаками: соціальні мережі для спілкування (Relationship networks), соціальні мережі для обміну медіа-контентом (Media sharing networks), соціальні мережі для відгуків і оглядів (Online reviews), соціальні мережі для колективних обговорень (Discussion forums), соціальні мережі для авторських записів (Social publishing platforms).

Згідно опитування Pew center [27] у вересні 2013 року, чверть Інтернет-користувачів пише свої коментарі анонімно. Користувачі молодшого віку (до 30 років) виявляють більше небажання пов'язувати своє реальне ім'я з онлайн-висловлюванням (до 40%). Анонімність Інтернет-коментування призвела до появи феномену, який психолог Джон

Салер назвав «ефектом мережевої розкутості». У момент, коли людина приховує свою ідентичність, вона також позбавляєтеся звичних стримувальних факторів своєї поведінки, що може створювати культуру агресії та знущання й унеможливлює осмислену дискусію. Зазвичай соціальні мережі, які дозволяють публікувати різноманітний контент без реєстрації або реєстрації з фіктивними обліковими записами, не мають жорстких обмежень на типи матеріалів для публікації, не несуть за нього відповідальності та не зацікавлені в модерації. Наприклад Tumblr (соціальна мережа для авторських записів, блогінгу і мікро-блогінгу, де користувачі створюють і публікують текстово-медійний контент), взагалі не має обмежень для контенту, тому виявлення образливого вмісту для цього виду соціальних мереж не є актуальним.

Соціальні мережі для обміну медіа-контентом, дає користувачам широкі можливості для обміну відео- та фото-контентом, зазвичай проводять модерацію не за текстовими повідомленнями, а за змістом відео- та фото-контенту відповідно. До таких соціальних мереж можна віднести Flickr, Instagram, YouTube, Vimeo, Vine, Snapchat. Дослідники Вісконсинського університету в Медісоні виявили, що манера висловлювання думок користувачами в секції коментарів відчутно впливає на ввічливість та адекватність наступних відгуків відвідувачів сайтів (незалежно від їх анонімності): чим грубіше коментарі під статтею, тим сильніше читачі починають розходитися в оцінках її змісту. Цей феномен вчені назвали «ефектом грубості» [28]. Аналогічно можна припустити, що відео та фото матеріали, що містять образливий вміст будуть провокувати появу агресивних коментарів, але на етапі модерації медіа матеріалів пост буде заблоковано, що відповідно призведи до блокування коментарів. У контексті даного дослідження цей вид соціальних мереж не розглядався, але автор вважає виявлення образливого вмісту у даних соціальних мережах перспективним напрямком досліджень, адже до будь-якого виду контенту можна додати опис чи назву, які міститимуть образливий вміст.

Найбільш цікавою для дослідження взаємозв'язків між користувачами автор вважає першу категорію соціальних мереж, а саме – соціальні мережі для спілкування, адже цей тип соціальних мереж є найпоширенішим і найбільш затребуваним видом соціальних мереж на сьогоднішній день. До цього типу відносяться Facebook, Вконтакте, Одноклассники, Linkedin. Даний формат соціальних мереж одним з перших запропонував користувачам створити безкоштовний персональний міні-сайт, який пізніше став відомий як профіль. Зазвичай користувачі асоціюють себе зі своїм профілем, а мережі взаємин намагаються запропонувати користувачам максимум можливостей в межах однієї платформи.

Умовно об'єднавши між собою найбільш популярні соціальні мережі, можна виявити наступні особливості взаємозв'язків між користувачами:

1. Фолловінг – це відслідковування появи нового контенту на сторінці особи на яку підписаний фоловер.
2. Особисті сторінки – це закриті або публічні сторінки з інформацією про власника сторінки та стрічкою з постами даного користувача.
3. Публічні групи – сторінки в соціальних мережах, які об'єднують певну спільноту людей, що зацікавлена в контенті, який генерує спільнота. Зазвичай дані сторінки є публічними.

На сьогоднішній день так як кожен користувач може поскаржитися на пост або коментар іншого користувача, що знижує якість виявлення образливого вмісту та спричиняє зменшення свободи слова в мережі Інтернет. Саме тому варто розмежовувати закриті персональні сторінки та публічні сторінки, на які можуть зайти діти або дуже вразливі люди. Варто враховувати дану особливість комунікації в соціальних мережах.

Отже, щоб врахувати зв'язки в соціальних мережах, пропонується ввести наступний тип вхідних даних:

{

autor: user

complainer: user

publication: private closed/public page,

another user's page,

community page

marked people: list of users

text: text for detection

}

Даний тип даних, дозволяє враховувати, автора повідомлення, користувача, що скаржиться, тип публікації: приватна закрита, чи приватна відкрита сторінка, повідомлення написане на сторінці іншого користувача чи публічній сторінці, список згаданих людей в повідомленні і відповідно сам текст повідомлення.

Враховування списоку згаданих в повідомленні людей дозволяє збільшити вплик скаржника, якщо він згаданий у повідомленні, що визначається, як повідомлення з образливим вмістом.

2.3 Розроблений метод визначення образливого вмісту в повідомленнях Інтернет-користувачів в соціальних мережах

Як було зазначено вище, існує низка особливостей, що характеризує спілкування Інтернет-користувачів в соціальних мережах. Автор вважає за доцільне враховувати такі характеристики повідомлень:

1. Наявність символів та цифр, що можуть призвести до заплутування сприйняття повідомлення – символи та цифри, що візуально сприймаються за існуючі літери, можуть негативно вплинути на коректність визначення наявності образливого вмісту повідомлення.
2. Контекст (на рівні повідомлень в соціальних мережах) – автор вважає, що врахування контексту сукупності повідомлень може

значно погіршити часові показники роботи методу з визначення образливого вмісту. Проте, абсолютне ігнорування контексту може спричинити зниження якості роботи методу. Саме тому пропонується врахувати контекст окремих повідомлень шляхом використання підходів до автоматизованого визначення образливого вмісту текстових даних, які працюють не з окремими словами чи фразами (наприклад, підходи, що базуються на пошуку ключових слів), а з вектором слів.

3. Наявність емодзі – з точки зору автоматизованого оброблення текстових даних емодзі – це по суті символи, що закодовані в юнікод. Наприклад, U+1F601 – це код смайлу емодзі. Звичайний класифікатор відкинув би ці дані на етапі передоброблення тексту як непотрібні символи, хоча насправді, дані символи впливають на сприйняття користувачем повідомлень, шляхом зміни їх емоційного забарвлення. Зважаючи на дані дослідження [24], автор вважає за доцільне додати можливість врахування емодзі при визначенні образливого вмісту повідомлень Інтернет-користувачів у соціальних мережах.
4. Контекст (на рівні зв’язків користувачів в соціальних мережах) – щоб врахувати зв’язки між користувачами в соціальних мережах при аналізі вмісту їх повідомлень, пропонується використати запропонований у попередньому пункті формат вхідних даних:

Врахування списку згаданих в повідомленні людей дозволяє збільшити вагомість скарги користувача соцмережі при прийнятті рішення алгоритмом щодо образливості контенту (якщо скаржник згаданий у повідомленні, що визначається як повідомлення з образливим вмістом, то такий пост потрібно заблокувати). Повернімося до прикладу «My neighbour is fucking cockroach.». Якби дане повідомлення будо написане наступним чином «My neighbour @user is fucking cockroach.», де @user – посилання в

соціальній мережі на акаунт іншого користувача, який є сусідом автора тексту і щодо якого написано дане образливе повідомлення, а також @user поскаржився на даний коментар, так як у ньому є образливий вміст, а саме особиста образа, то таке повідомлення мало б бути негайно видалене з соціальної мережі.

Визначимо тепер, на якому етапі процесу детекції образливого вмісту текстових даних можна врахувати зазначені вище характерні риси повідомлень Інтернет-користувачів.

Умовно даний процес можна розділити на три етапи: передоброблення тексту, класифікацію підготовленого тексту та аналіз результатів класифікації. Автор вважає за доцільне здійснити модифікацію етапу передоброблення тексту, щоб врахувати спецсимволи та емодзі, а також ввести новий формат вхідних даних, який було описано вище, використання якого дозволить враховувати контекст та стосунки між користувачами соціальних мереж.

Передоброблення тексту пропонується проводити безпосередньо над текстовим повідомленням за допомогою класичних методів, які модифікуються шляхом додавання двох допоміжних кроків.

По-перше, вхідний текст має бути перевіреним на наявність в ньому символів, що можуть бути використаними для заплутування написання слів з метою обходу визначення образливого вмісту класифікатором. Наприклад, такі символи як «99» мають бути замінені на «gg», «@» на «a», «$$» на «ss» та тому подібне. Даний крок ускладнить вживання образливих слів та дозволить покращити точність роботи класифікатора.

По-друге, автор вважає за доцільне перетворювати емодзі-символи на текстові дані, що дозволить не втрачати смислового та емоційного навантаження на текст. Існують стандартизовані таблиці емодзі-символів з кодами та значеннями, що несе в собі той чи інший код емодзі. Використання тлумачень кодів емодзі дозволить легко перетворити їх на

слова або словосполучення та використати при класифікації на відміну від звичайного передоброблення, де дані символи було б відкинуто.

Визначення образливого вмісту текстових повідомлень, що пройшли передоброблення, пропонується здійснювати на основі класичних методів визначення образливого вмісту текстових даних (наприклад, за допомогою Наївного баєсівського класифікатора). Жодних змін тут не потрібно.

Результат, наданий класифікатором (ймовірність віднесення до класу «образливий» текст), буде коригуватися на фінальному етапі з урахуванням зв’язків між користувачами в соціальній мережі та місцем публікації. Мається на увазі, якщо текст є образливим, але він був опублікованим на приватній сторінці і не має посилань на інших користувачів, то дане повідомлення не буде блокуватися. Якщо на ньому було відмічено певних людей, тоді текст має бути додатково перевіреним (наприклад, людиною-модератором). Проте, якщо даний текст опубліковано на публічній сторінці, то його має бути заблоковано. Даний підхід дозволяє убезпечити користувачів від образливого контенту, а з іншого боку – зменшує рівень цензури та дозволяє вільно висловлюватися. Для формалізації етапу аналізу результатів було побудовано відповідне дерево рішень (рис. 2.3.1).

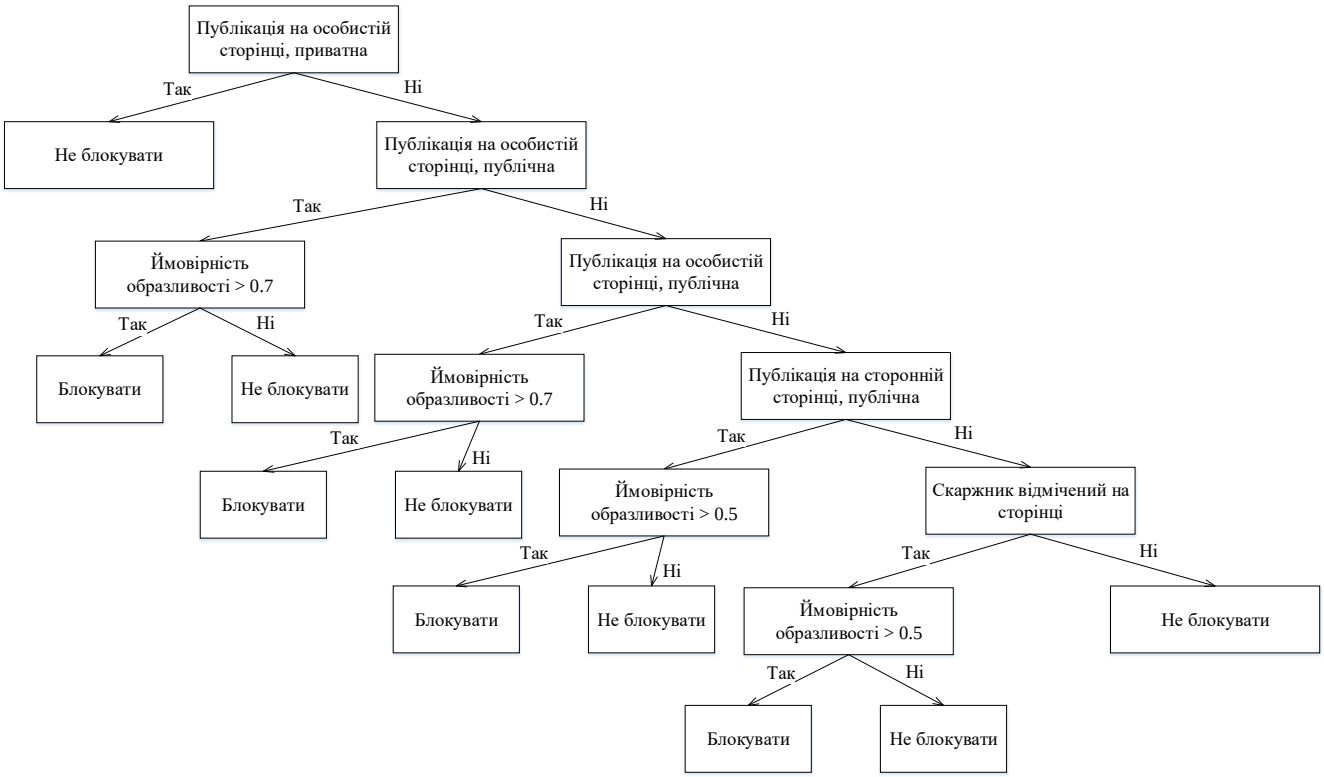

Рис. 2.3.1. Дерево рішень етапу аналізу результатів

Отже, у запропонованому автором методі можна виділити такі етапи визначення образливого вмісту повідомлень користувачів соціальних мереж (рис. 2.3.2):

1. передоброблення тексту:
    a. із заміною спецсимволів на літери, які візуально схожі між собою;
    b. з перетворенням емодзі на текстові дані;
2. визначення образливого вмісту відібраного повідомлення за допомогою класичних алгоритмів визначення образливого вмісту природномовних текстових даних;
3. формування результатів з урахуванням оцінки класифікатора та зв’язків Інтернет-користувачів в соціальній мережі.

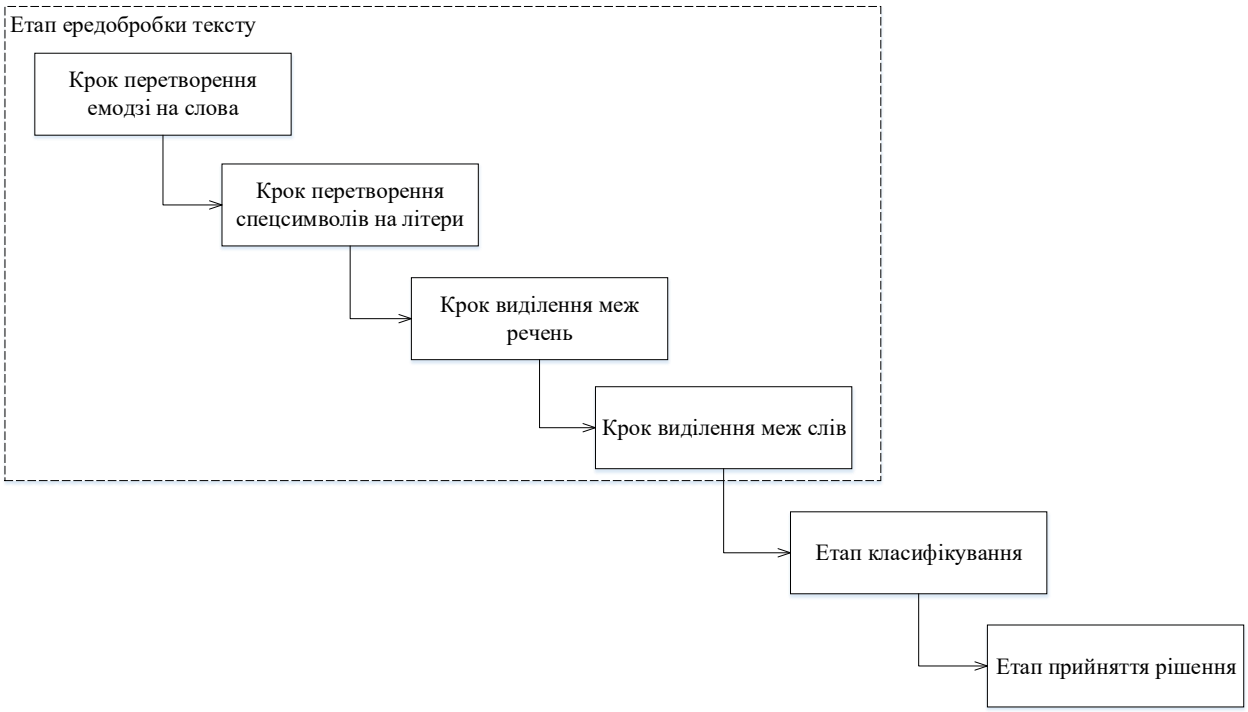

Рис. 2.3.2. Послідовність виконання кроків методу

2.4 Висновки

В рамках дослідження, описаного в даному розділі, проведено вивчення специфіки публікацій Інтернет-користувачів у соціальних мережах. Розглянуто існуючі механізми визначення образливого вмісту в соціальних мережах, визначено їх ключові недоліки, а саме відсутність врахування контексту на рівні стосунків між користувачами соціальних мереж, вразливість до заплутування слів та фраз, відсутність механізму обробки емодзі символів. На базі визначених особливостей та недоліків існуючих рішень сформовано перелік характеристик повідомлень Інтернет-користувачів в соціальних мережах, які можуть бути враховані під час визначення образливого вмісту, а саме: наявність символів, що можуть призвести до заплутування сприйняття слів і фраз; контекст (на рівні повідомлень в соціальних мережах); контекст (на рівні зв’язків користувачів в соціальних мережах); наявність емодзі.

Для врахування особливостей стосунків між різними групами Інтернет-користувачів в соціальних мережах запропоновано новий формат

вхідних даних для методу визначення образливого вмісту, що дозволяє не втратити контекст на рівні зв’язків користувачів в соціальних мережах.

Проведено аналіз процесу детекції образливого вмісту текстових даних з точки зору можливості урахування в ньому специфіки повідомлень користувачів соціальних мереж, а саме запропоновано модифікацію етапів передоброблення тексту та аналізу результатів класифікації. Розроблено метод визначення образливого вмісту повідомлень Інтернет-користувачів в соціальних мережах на основі комплексного урахування характеристик повідомлень в соціальних мережах та специфіки стосунків між різними групами відвідувачів соціальних мереж.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗОВАНОГО ВИЗНАЧЕННЯ ОБРАЗЛИВОГО ВМІСТУ ТЕКСТОВИХ ПОВІДОМЛЕНЬ В СОЦІАЛЬНИХ МЕРЕЖАХ

3.1 Опис засобів розробки програмного забезпечення

Програмна реалізація застосунку для автоматизованого визначення образливого вмісту текстових повідомлень в соціальних мережах виконана за допомогою мови програмування C# (C Sharp) на платформі .NET. Для реалізації програмного забезпечення використовувалось середовище розробки – Visual Studio 2017 Community edition.

Нижче наведений короткий опис використаних засобів.

1. *Мова програмування C#*

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET, яка була розроблена в на зламі тисячоліть інженерами компанії Microsoft. На сьогоднішній день дана мова програмування посідає 5-те місце в рейтингу найбільш популярних мов програмування [29]. За допомогою мови C# можна створювати різноманітних безпечні та потужні застосунки, які виконуються в середовищі .NET Framework. Мова програмування C# дозволяє реалізовувати широкий спектр різноманітного програмного забезпечення: від невеличких навчальних програм до великий інтерпрайзних проектів.

C# має C-подібний синтаксис, завдяки чому програми на C# візуально схожі до мов програмування C++ і Java. C# – це строго типізована мова програмування, а значить компілятор на початку компіляції знає, який тип має змінна. Також мова має наступні функціональні можливості:

- не підтримує множинного наслідування,
- підтримує поліморфізм,
- перевантаження операторів,
- атрибути,

- події,
- властивості,
- вказівники на функції-члени класів,
- коментарі у форматі XML,
- має кортежний тип данних – тупл (від англ. tuple).

C# – об’єктно-орієнтована мова, а саме тому реалізує основні принципи ООП:

- інкапсуляція – змінні і методи інкапсулюються в межах визначеного класу, до якого вони належать;
- наслідування – клас (нащадок) може наслідуватися безпосередньо з одного батьківського класу.
- поліморфізм – клас може реалізовувати будь-яку кількість інтерфейсів.

У мові C# існує 2 типи: тип значення (від англ. value) та тип посилання (від англ. reference). Перший тип реалізується за допомогою структур, а тип посилання реалізується у вигляді класів. Структури визначаються за допомогою ключового слова struct. Структури використовують більшу частину того ж синтаксису, що й класи, проте вони більш обмежені в порівнянні з ними, наприклад, може реалізувати інтерфейси, але не підтримує наслідування. Основною відмінністю між класом і структурою є те, що структури зберігаються в стеку, а класи лежать в кучі (від англ. heap).

В 7 версії мови додалося багато корисних функціонльних можливостей, а саме: out-змінні, паттерн зіставлення з шаблоном (pattern matching), шаблони з is, кортежі, розпакування кортежів (деконструктор), локальні функції, розширення списку типів, що повертаються асинхронними методами та інше.

Мова C# дозволяє дуже швидко та просто розробляти програмні застосунки різної складності, завдяки кільком інноваційним конструкціям

мови а також так званому «синтаксичному цукру», до числа яких входять наступні:

- інкапсульовані сигнатури методів – делегати, дозволяють безпечні типізовані повідомлення про події, надають механізм пізнього зв’язування в .NET;
- властивості, забезпечують безпечний доступ до закритих змінних полів класу;
- атрибути надають потужний засіб для зв’язування метаданих або декларативною інформації з кодом (збірки, типи, методи, властивості і т. д.) під час виконання;
- вбудовані коментарі XML документації;
- LINQ, що пропонує нативні можливості виконання запитів;
- перевантаження операторів (в тому числі операторів явного і неявного приведення типу) – ключове слово operator оголошує функцію, яка вказує, що означає operator-symbol при застосуванні до примірника класу. Це дає оператору більш одного значення, а компілятор розрізняє різні значення оператора, перевіряючи типи його операндів;
- події дозволяють класу або об’єкту повідомляти інші класи або об’єкти про виникнення будь-яких ситуацій. Клас, що надсилає (або породжує) подію, називається видавцем, а класи, які отримують (або ті, що опрацьовують) події, називаються підписниками;
- узагальнені типи і методи – інкапсулюють операції, які не належать до конкретного типу даних;
- ітератори призначення для проходу по колекціям;
- анонімні функції з підтримкою замикань;
- виключені ситуації.

Переваги мови C#:

– мова постійно розвивається;
– розроблялася паралельно з новим каркасом Framework .Net і повністю відповідає всім його потенціалам;
– компілятор з відкритим кодом;
– необмежені можливість успадкування та універсалізації;
– велика база документації;
– зберігши основні риси своїх прабатьків, C# є більш надійною і простою в порівнянні з попередниками;
– автоматичне керування пам’яттю;
– ефективний і надійний код робить проект C# успішним і прогресивним [30];
– наявні інструменти для рефакторингу та діагностики програмного забезпечення;
– наявні елементи функціонального програмування.

2. *Платформа .NET*

Microsoft .NET – крос-платформова програмна технологія для створення широкого спектру програмних застосунків, розроблена корпорацією Майкрософт, що працює переважно на Microsoft Windows.

Microsoft .NET включає в себе бібліотеку великого класу з ім’ям Framework Class Library (FCL) та забезпечує сумісність мов (кожен мова може використовувати код, написаний іншими мовами) на кількох мовах програмування. Програми, написані для .NET Framework, виконуються у програмному середовищі (на відміну від апаратного середовища), яке називається Common Language Runtime (CLR) – це віртуальна машина програми, яка надає послуги, такі як безпека, керування пам’яттю та обробка виключень. Комп’ютерний код, написаний з використанням .NET Framework, називається «керованим кодом». FCL та CLR є двома основними складовими частинами, китами на якиї стоїть .NET Framework.

FCL забезпечує користувальницький інтерфейс, доступ до даних, підключення до бази даних, криптографію, розробку веб-додатків, цифрові

алгоритми та мережеві зв’язки. Програмісти створюють програмне забезпечення, об'єднуючи їх вихідний код з .NET Framework та іншими бібліотеками. Фрейворк призначений для використання більшістю нових додатків, створених для платформи Windows. Корпорація Майкрософт також випускає інтегроване середовище розробки, головним чином для .NET програмного забезпечення, яке називається Visual Studio, яке детальніше розглянемо трошки пізніше.

Нещодавно Windows випустили кросплатформену версію фрейворку. .NET Core – це вільна і відкрита версія керованого програмного забезпечення для Windows, macOS та Linux [31]. Він складається з CoreCLR, повної реалізації CLR, віртуальної машини, яка управляє виконанням програм .NET.

.NET Core також включає CoreFX, який є частковою вилкою FCL [32].

.NET Core підтримує чотири крос-платформних сценаріїв: веб-додатки ASP.NET Core, додатки командного рядка, бібліотеки та універсальні додатки для платформ Windows. .NET Core, на відміну від .NET Framework не реалізує Windows Forms або WPF, які надають стандартний графічний інтерфейс для настільних програм у Windows [33] [34].

.NET Core, як і .NET Framework, також є модульним, що означає, що замість збірок розробники працюють з NuGet-пакетами. На відміну від .NET Framework, який обслуговується за допомогою Windows Update, .NET Core покладається на свого менеджера пакетів для отримання оновлень [33][34].

Основні переваги платформи .NET:

- компонентно-орієнтований підхід до проектування та реалізації програмного забезпечення полягає у принциповій можливості створення незалежних складових програмного

забезпечення з уніфікованою інтерфейсною частиною для багаторазового повторного і розподіленого використання [35];

- сувора ієрархічність організації просторів для типів, класів та імен сутностей програми дозволяє стандартизувати і уніфікувати реалізацію;
- платформа забезпечує одночасну підтримку проектування та реалізації програмного забезпечення з використанням різних мов програмування;
- універсальний інтерфейс .NET Framework забезпечує інтегроване проектування та реалізацію компонентів додатків, розроблених відповідно до різних підходів до програмування;

3. *Середовище розробки Visual Studio 2013*

Microsoft Visual Studio – це інтегроване середовище розробки (IDE) від Microsoft, що використовується для розробки комп'ютерних програм, а також веб-сайтів, веб-застосунків, веб-сервісів та мобільних додатків. Visual Studio використовує платформи для розробки програмного забезпечення Microsoft, такі як Windows API, Windows Forms, Windows Presentation Foundation, Windows Store. Microsoft Visual Studio дозволяєстворювати як власний (некерований) код, так і керований код.

Visual Studio включає в себе редактор коду, що підтримує IntelliSense (компонент завершення коду), а також рефакторинг коду. IntelliSense – технологія автодоповнення Microsoft, найбільш відома в Microsoft Visual Studio. Дописує назву функції при введенні початкових літер. Крім прямого призначення, IntelliSense використовується для доступу до документації та для усунення неоднозначності в іменах змінних, функцій і методів, використовуючи рефлексію. Наявний вбудований відладчик (зневаджувач, дебагер), який дозволяє дуже зручно проводити зневаджування коду, навіть асинхронного. Інші вбудовані інструменти включають програму для кодування, конструктор форм для побудови графічних інтерфейсів, веб-дизайнер, дизайнер класів та дизайнер схеми

баз даних. Visual Studio може приймати плагіни, які покращують функціональність практично на всіх рівнях, включаючи підтримку систем керування версіями (наприклад, Git).

Visual Studio підтримує 36 різних мов програмування. Вбудовані мови включають C, [36] C ++, C ++ / CLI, Visual Basic. NET, C #, F #, [37] JavaScript, TypeScript, XML, XSLT, HTML та CSS. Підтримка інших мов, таких як Python, [38] Ruby, Node.js та M, серед інших, доступна через плагіни. Раніше підтрумувалися також Java та J#.

Найбільшою перевагою Visual Studio є те, що Community Edition, доступне безкоштовно.

Visual Studio 2017 пропонує нові функції, такі як підтримка EditorConfig, підтримка NGEN, .NET Core та набір інструментів Docker (Preview), Xamarin 4.3 (Preview) [39]. Він також має редактор XAML (декларативна мова розмітки), покращений IntelliSense, юніт тестування на льоту (live unit testing), вдосконалення зневаджувача та покращення продуктивністі IDE.

3.2 Архітектура розробленого програмного забезпечення

Розроблену систему можна розділити на десять модулів, зображених на рис. 3.2.1:

1. Модуль перетворення емодзі на слова – надає можливість заміни кодів емодзі на слова, що описують даний тип емодзі. Це дозволяє уникненню видалення додаткової смислової навантаженості та тональності, яку несуть в собі емодзі. На вхід приймає весь вхідний текст повідомлення користувача. У вхідному тексті за наявності знаходить емодзі-код та конвертує його у відповідне слово або словосполучення. На вихід віддає текс без емодзі кодів. Отримані результати передає наступному етапу передобробки (конкретний модуль не вказується через наявність функціональних можливостей, які надає модуль конфігурації, про який буде йтися далі).

2. Модуль перетворення спецсимволів на літери – надає можливість заміни спецсимволів на літери, які візуально схожі на літери і можуть призводити до заплутування класифікатора. На вхід приймає весь вхідний. Здійснює перетворення спецсимволів та знаків пунктуації на візуально схожі літери. На вихід віддає текст без символів, що можуть змінити значення слів та вплинути на якість роботи класифікатора. Отримані результати повертає наступному модулю передобробки вхідного тексту повідомлення.

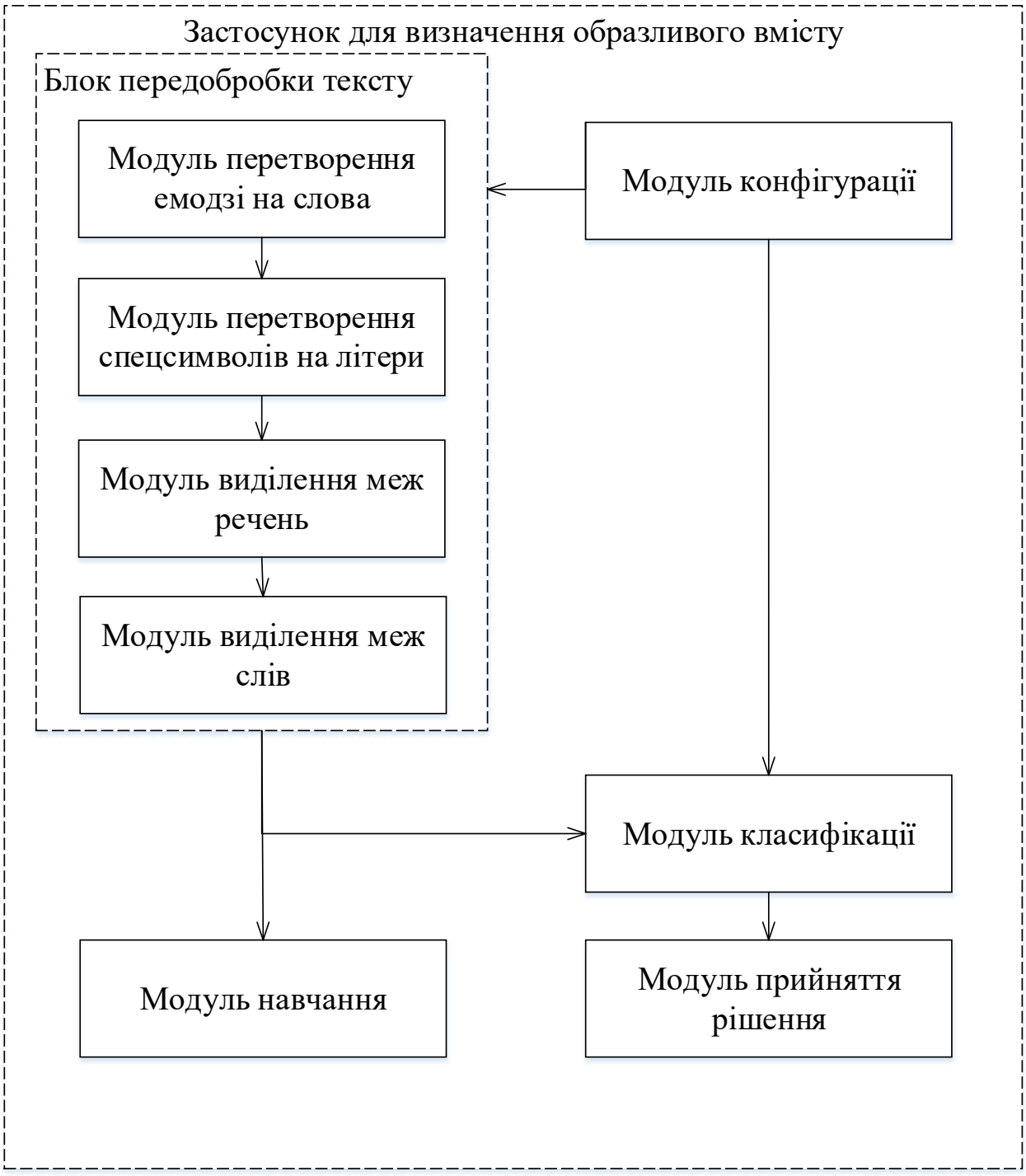

Рис. 3.2.1. Схема модулів програми

3. Модуль виділення меж речень – надає можливість розбиття вхідного тексту повідомлення на речення. На вхід приймає вхідний текст повідомлення. Здійснює розбиття вхідного тексту на синтаксичні одиниці – речення. На вихід віддає масив, що складається з речень вхідного тексту. Результати роботи цього модуля використовуються модулем виділення меж слів.

4. Модуль виділення меж слів – надає можливість розбиття вхідного масиву речень на слова. На вхід приймає масив речень. Здійснює розбиття кожної вхідної синтаксичної одиниці на лексичну – слова. На вихід віддає масив, що складається з слів вхідного тексту. Результати роботи цього модуля використовуються модулем класифікації та можуть бути використаними модулем навчання.

Модуль перетворення емодзі на слова, модуль перетворення спецсимволів на літери, модуль виділення меж речень та модуль виділення меж слів можна об’єднати в блок передобробки вхідного тексту повідомлення, оскільки всі вони здійснюють підготовчі роботи над текстом для його подальшої класифікації.

5. Модуль конфігурації – задає параметри класифікації. Визначає, які модулі передобробки текстових будуть задіяні під час класифікації. Може опціонально вимикати кроки передобробки наприклад, модуль перетворення емодзі на слова або модуль перетворення спецсимволів на літери, або вимкнути обидва модулі одночасно.

6. Модуль навчання – надає можливість створити моделі для навчання класифікатора на основі попередньо розмічених текстів. На вхід приймає масив слів з блоку передобробки тексту, а також значення класу, до якого належить досліджуваний об’єкт. На вихід віддає модель, на основі якої буде працювати класифікатор.

7. Модуль класифікації – надає можливість виявлення у вхідному тексті повідомлення образливого вмісту. На вхід приймає масив слів – попередньо передоброблений текст. За допомогою наївного баєсового класифікатора здійснює класифікацію вхідного тексту. На вихід віддає словник ймовірностей приналежності вхідного тексту до одного з двох класів: нейтральний текст чи образливий. Результати роботи цього модуля використовуються модулем прийняття рішень.

8. Модуль прийняття рішень – надає можливість визначення рекомендації до блокування вхідного тексту. словник ймовірностей

приналежності вхідного тексту до одного з двох класів: нейтральний текст чи образливий. За допомогою розширеного формату вхідних даних приймає рішення щодо рекомендації до блокування тексту. На вихід віддає рекомендацію щодо блокування відного тексту чи ні.

Розглянемо більш детальніше частину архітектури розробленого програмного забезпечення. На рис. 3.2.2 зображена загальна діаграма класів системи визначення образливого вмісту.

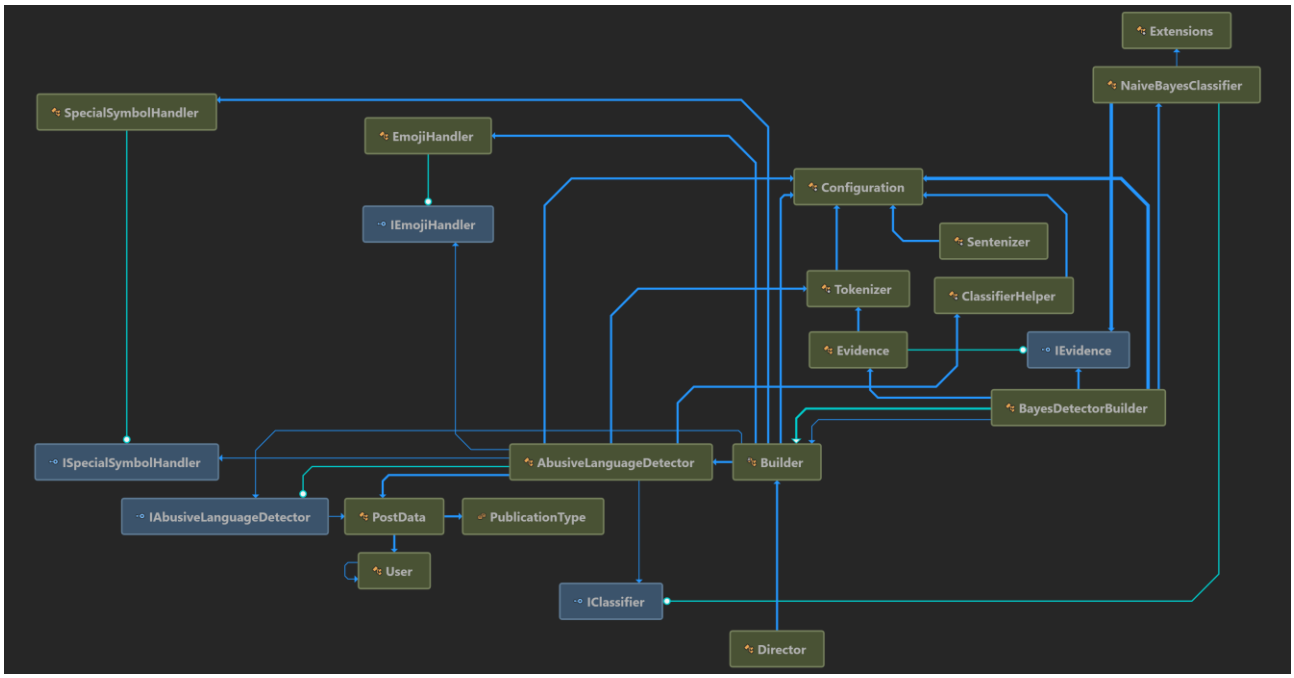

Рис. 3.2.2. Загальна діаграма класів системи визначення образливого вмісту

Для кожного великого логічного модуля створено окремі простори імен (Namespace). На рис. 3.2.3 зображена загальна діаграма просторів імен системи визначення образливого вмісту.

Для кожного великого логічного модуля створено окремі простори імен (Namespace). На рис. 3.2.3 зображена загальна діаграма просторів імен системи визначення образливого вмісту.

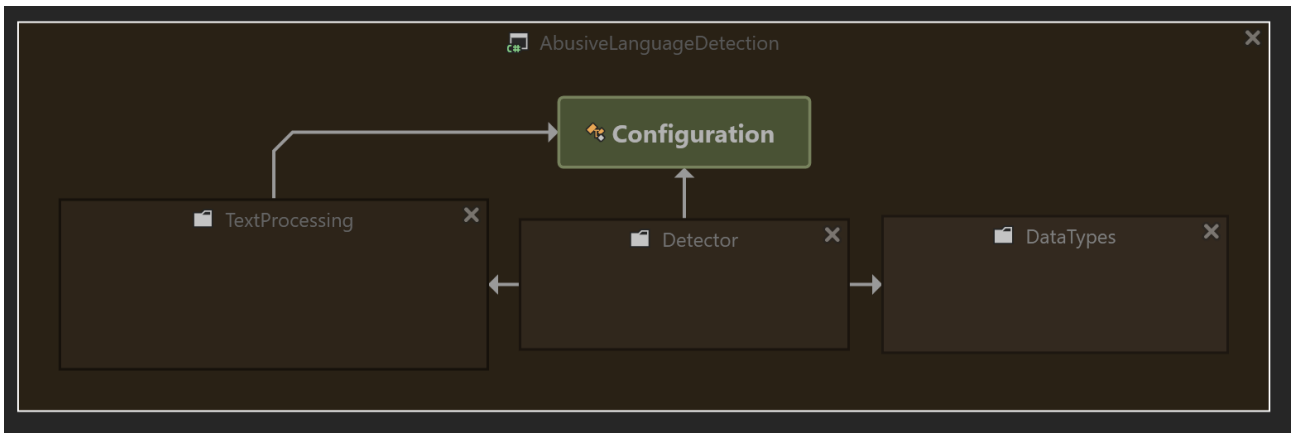

Рис. 3.2.3. Загальна діаграма просторів імен системи класифікації коротких новинних текстів

3.3 Особливості реалізації програмного застосунку

На основі наведених теоретичних засад та розробленого методу визначення образливого вмісту в повідомленнях Інтернет-користувачів було розроблено програмний засіб автоматизованого визначення образливого вмісту в повідомленнях Інтернет-користувачів. Програмне забезпечення реалізоване з використанням принципів ООП і має об’єктно-орієнтовану структуру.

В якості мови вхідних текстів обрано англійську. Це пов’язано з тим, що англійська мова є дуже розповсюдженою (посідає перше місце серед розповсюдженості мов, нею володіють 1,500 мільйони людей [40]).

Для спрощення реалізації даного програмного забезпечення в основу його структурної організації було покладено підхід до побудови програмного забезпечення з використанням шаблонів проектування. Необхідність використання шаблонів зумовлена високою динамічністю та модульністю будови програми, оскільки користувач може самостійно обирати параметри за допомогою конфігуратора, за якими проводитиме передобробку.

Шаблони проектування в свою чергу сприяють забезпеченню високого рівня гнучкості розроблюваного програмного забезпечення щодо внесення або видалення з нього елементів, спрощенню подальшої

підтримки програмного продукту, завдяки можливості змінювати її частини незалежно одна від одної тощо. Для реалізації даного застосунку було використано шаблон «Будівельник» (builder) [41].

Крім того під час розробки програмного забезпечення, автор керувався принципами об’єктно-орієнтованого дизайну, а саме SOLID – це абревіатура складена з перших літер п’яти базових принципів об'єктно-орієнтованого програмування та дизайну запропонована Робертом Мартіном.

Принципи SOLID використовуються для дизайну та розробки таких програмних систем, які, з великою ймовірністю, зможуть тривалий час розвиватися, розширятися і підтримуватися, а з врахуванням того факту, як планується просувати даний програмний застосунок на ринок використання вищезгаданих принципів є дуже актуальним.

Для забезпечення максимальної принципів SOLID під час розробки було використано абстракції та інтерфейси, які потім можна легко замінити конкретними реалізаціями. Використання абстракції та інтерфейсів знизити рівень зв’язності коду, що спрощує його підтримку та подальше розширення для розробників та нових людей команди.

Для визначення образливого вмісту був реалізований детектор образливого вмісту, який описує відповідний програмний інтерфейс.

Лістинг 1. Інтерфейс детектора образливого вмісту

```
public interface IAbusiveLanguageDetector
{
     bool IsAbusiveLanguage(PostData postData);
}
```

Також використовуються зав’язка на абстракції в реалізації AbusiveLanguageDetector. В реалізації дано методу використаний підхід до побудови програмного забезпечення, а саме ін’єкції залежності.

Ін’єкція залежності (від англ. Dependency injection, DI) – це шаблон проектування програмного забезпечення, що передбачає надання зовнішньої залежності програмному компоненту, використовуючи

«інверсію управління» (від англ. Inversion of control, IoC) для отримання залежностей.

Існує три найбільш поширені форми реалізації ін'єкція залежностей:

- ін'єкція в конструктор,
- ін'єкція у властивість,
- ін'єкція в метод.

Тобто при ін'єкції залежностей об'єкт є пасивним і не вживає взагалі ніяких кроків для з'ясування залежностей, а надає для цього сеттери і / або приймає своїм конструктором аргументи, за допомогою яких впроваджуються залежності [42].

У даній реалізації була обрана ін'єкція через конструктор.

Лістинг 2. Використання інтерфейсів для створення залежностей в коді

```
        private IClassifier _classifier;
        private ISpecialSymbolHandler _specialSymbolHandler;
        private IEmojiHandler _emojiHandler;

        public AbusiveLanguageDetector(IClassifier classifier, ISpecialSymbolHandler
specialSymbolHandler, IEmojiHandler emojiHandler)
        {
            _classifier = classifier;
            _specialSymbolHandler = specialSymbolHandler;
            _emojiHandler = emojiHandler;
        }
```

Контракти для методів передобробки, а саме модуль перетворення емодзі на слова та модуль перетворення спецсимволів на літери, мають наступний вигляд.

Лістинг 3. Інтерфейс IEmojiHandler

```
public interface IEmojiHandler
{
        string HandleEmoji(string content);
}
```

Лістинг 4. Інтерфейс ISpecialSymbolHandler

```
public interface ISpecialSymbolHandler
{
        string HandleSpecialSymbol(string content);
}
```

Загалом блок передобробки тексту реалізовано за допомогою методу PreProcessing, який в залежності від конфігурації виконує етап

перетворення емодзі на слова або ні, та відповідно етап перетворення спецсимволів на літери.

Лістинг 5. Код передобробки тексту

```
private List<string> PreProcessing(string data)
{
	if (Configuration.UseEmojiHandler)
	{
		data = HandleEmoji(data);
	}

	if (Configuration.UseSpecialSymbolHandler)
	{
		data = HandleSpecialSymbols(data);
	}

	List<string> content = Tokenizer.TokenizeNow(data).ToList();

	return content;
}
```

Для врахування контексту на рівні стосунків між користувачами соціальної мережі було впроваджено спеціальний тип даних PostData. Відображення типу публікації відбувається за допомогою відповідного перелічуваного типу даних (від англ. enum) в мові програмування C#.

Лістинг 6. Типи публікацій

```
public enum PublicationType
{
	PrivatePersonalPage,
	PublicPersonalPage,
	AnotherUserPage,
	CommunityPage
}
```

В класі PostData є поле, яке відповідає за дані про користувача, який створив публікацію, доступ до якого реалізовано через властивість.

Лістинг 7. Дані про користувача, який створив публікацію, в класі PostData

```
private User _author;
public User Author
{
    get => _author;
    private set { _author = value; }
}
```

Також клас PostData відображає дані про користувача, який поскаржився на публікацію, доступ до якого реалізовано через відповідну властивість.

Лістинг 8. Дані про користувача, який поскаржився на публікацію, в класі PostData

```
private User _complainer;
public User Complainer
{
    get => _complainer;
    private set { _complainer = value; }
}
```

Лістинг 9. Тип публікаціїв класі PostData

```
private PublicationType _publicationType;
public PublicationType PublicationType
{
    get => _publicationType;
    private set { _publicationType = value; }
}
```

Крім того, клас PostData зберігає список користувачі, які були відмічені в публікації, доступ до якого реалізовано через відповідну властивість.

Лістинг 10. Список користувачів, відмічених в публікації, в класі PostData

```
private List<User> _markedPeople;
public List<User> MarkedPeople
{
    get => _markedPeople;
    set => _markedPeople = value;
}
```

Також клас PostData імплементиє низку методів, що дозволяють здійснити наступні перевірки:

- Метод bool IsComplainerMarked() – перевіряє чи користувач, який поскаржився на публікацію, наявний в списку наявних в публікації користувачів.
- Метод bool IsPrivatePublication() – перевіряє, чи дана публікація розміщена на особистій приватній сторінці користувача, що написав текст публікації.
- Метод IsPersonalPublicPublication() – перевіряє, чи дана публікація розміщена на особистій публічній сторінці користувача, що написав текст публікації.

- Метод IsPublicOnOtherPage() – перевіряє, чи дана публікація розміщена на публічній сторінці іншого користувача або публічної спільноти.

Безпосередньо за класифікацію відповідає клас Classifier, який імплементує сигнатуру інтерфейсу IClassifier. В Classifier виконана реалізація класифікації образливості тексту за допомогою наївного баєсівського класифікатора.

Лістинг 11. Інтерфейс IClassifier

```
public interface IClassifier
{
	IDictionary<string, double> Classify(List<string> content,
HashSet<string> wordsToIgnore);
}
```

Метод IsAbusiveLanguage раалізує метод визначення образливого вмісту, шляхом виклику передобробки вхідного тексту, його класифікації та методу прийняття рішення.

Лістинг 12. Код визначення образливого вмісту

```
public bool IsAbusiveLanguage(PostData postData)
{
	List<string> content = PreProcessing(postData.Text);

	IDictionary<string, double> classificationResult =
_classifier.Classify(content, ClassifierHelper.ExcludeList);

	return MakeDecision(postData,
classificationResult[Configuration.AbusiveClass]);
}
```

Для спрощення побудови об’єктів було використано шаблон будівельник (від англ. Builder).

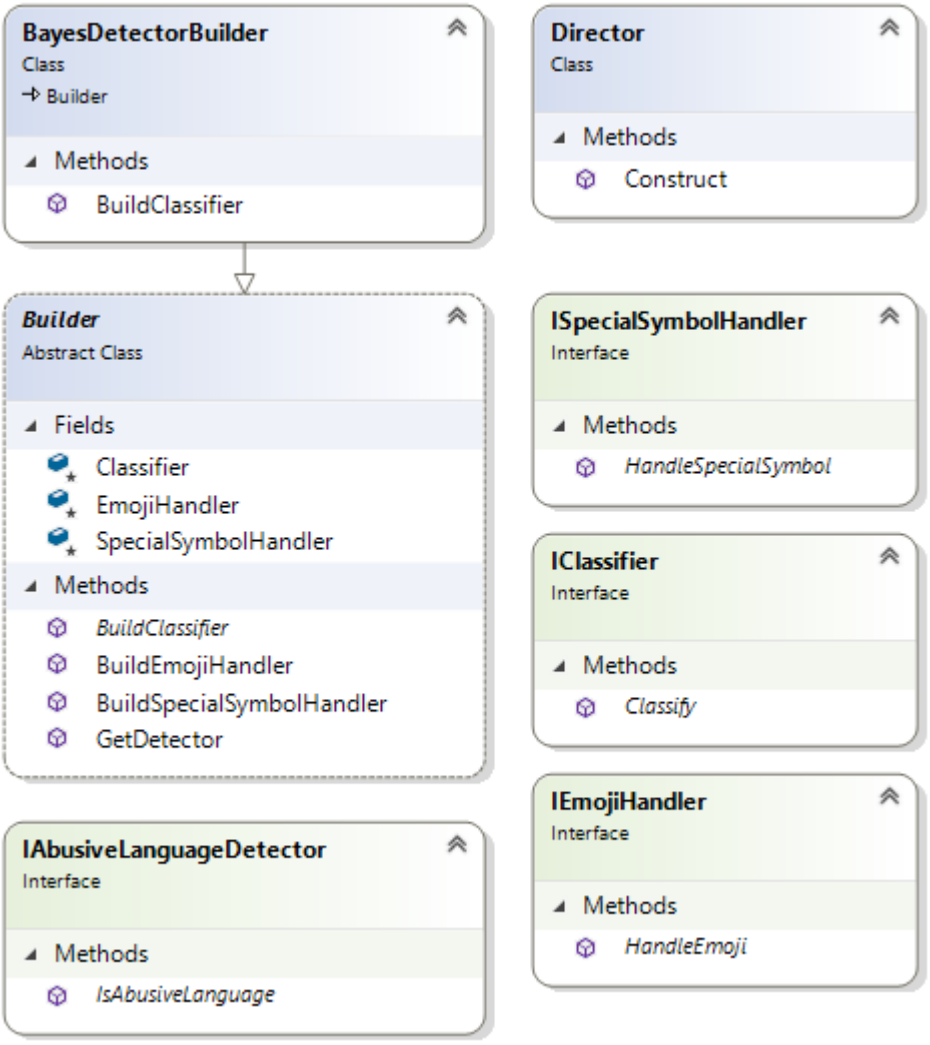

Рис. 3.3.1. Реалізація шаблону будівельник

Будівельник – це породжуючи шаблон дизайну програмного забезпечення, призначений для забезпечення гнучкого рішення різних задач створення об’єктів в об’єктно-орієнтованому програмуванні. Метою шаблону дизайну будівельник є відокремлення конструкції складного об’єкта від його представлення. Таким чином, один і той же процес будівництва може створити різні уявлення [43]. Цей паттерн належить до Gang of Four моделей дизайну.

Для спрощення керування конфігурацією реалізованої бібліотеки було створено статичний клас Configuration, який приймає на вхід всі директорії з необхідними даними для класифікації, а також визначає, чи необхідно викликати перетворення емодзі кодів на слова та заміни спецсимволів та знаків пунктуації на літери.

Лістинг 13. Код статичного класу Configuration

```
public static class Configuration
{
	public static string BaseFolder =
		@"\AbusiveLanguageDetection\Repo";
	public static string ExcludedTokensList = "ExcludedTokens.txt";
	public static string Sentenaizer = "en-sent.bin";
	public static string Tokeneizer = "en-token.bin";
	public static string SpecialSymbols = "SpecialSymbols.csv";

	public static string NeutralClass = "Neutral";
	public static string AbusiveClass = "Abusive";

	public static string LabeledData = "LabeledData.csv";

	public static bool UseEmojiHandler = true;
	public static bool UseSpecialSymbolHandler = true;
}
```

3.4 Висновки

У розділі наведено особливості реалізації програмних блоків та модулів, продемонстровано доцільність використання шаблонів проектування таких як «Будівельник», показано особливості їх реалізації при розробці даного програмного забезпечення. Продемонстровано реалізацію принципів ООП та дотримання принципів об'єктно-орієнтованого дизайну, а саме SOLID.

Розроблене програмне забезпечення призначене для автоматизованого визначення образливого вмісту повідомлень Інтернет-користувачів на основі запропонованого в роботі методу автоматизованого визначення образливого вмісту текстових повідомлень в соціальних мережах.

Розроблена система надає можливості налаштовувати параметри визначення образливого вмісту в залежності від потреб користувача.

Завдяки широкому застосуванню абстракцій архітектура розробленого програмного забезпечення дозволяє з легкістю додавати реалізацію інших методів визначення образливого вмісту, збільшувати кількість застосовуваних класифікаторів, а також змінювати, видаляти чи додавати певні кроки до вже реалізованого методу.

Реалізований програмний продукт може бути з легкістю інтегрованим у інші системи, легко працювати з будь-яким інтерфейсом чи програмним забезпеченням, як його частина.

4 АНАЛІЗ ЕФЕКТИВНОСТІ МЕТОДУ АВТОМАТИЗОВАНОГО ВИЗНАЧЕННЯ ОБРАЗЛИВОГО ВМІСТУ ТЕКСТОВИХ ПОВІДОМЛЕНЬ В СОЦІАЛЬНИХ МЕРЕЖАХ

Для аналізу ефективності запропонованого методу автоматизованого визначення образливого вмісту текстових повідомлень в соціальних мережах було використано власноруч розмічений набір з 300 повідомлень з соціальної мережі Twitter (твітів). Кожен твіт було віднесено до однієї з категорій:

1. HateSpeech.
2. OffensiveLanguage.
3. Neither.

У випадку цього даного дослідження перші дві категорії розглядалися, як образливий вміст, а третя – нейтральний.

Для дослідження ефективності результатів даний набір було двічі поділено на частини:

- 66.6% набору (200 твітів для навчання) та 33.3% (100 твітів) набору для класифікації.
- 90% набору (270 твітів для навчання) та 10% (30 твітів) набору для класифікації.

І після цього виконано аналіз за допомогою метрик, які широко застосовуються при оцінці точності роботи класифікатора. При виконанні одного тесту було використано звичайний наївний Баєсівській класифікатор, а при другому – використовувався запропонований метод.

4.1 Використані метрики оцінки

4.1.1 Частка правильно класифікованих об'єктів (Accuracy)

Найпростіша з застосовуваних метрик. Відображає кількість документів, які було віднесено до правильного класу.

$$Acc = \frac{Cor}{N},$$

де *Cor* – кількість правильно класифікованих документів вибірки, а N –розмір вибірки.

Наприклад, в нашій вибірці було 100 твітів, 70 з них наш класифікатор відніс до класу, який співпадає з визначенням експерту. Значить частка правильно класифікованих об’єктів нашого класифікатора – 0.7.

Проте, у цієї метрики є одна особливість яку необхідно враховувати. Вона приймає, що всі документи мають однакову вагу, що може бути не коректно в разі якщо розподіл документів в навчальній вибірці сильно зміщений в сторону якогось з класів. В цьому випадку в класифікатора є більше інформації для навчання певного класу і він буде краще працювати саме для визначення документів даного класу. На практиці це призводить до того, що ви маєте високу точність, але тільки з однім з класів, що при реально використанні може бути критично.

Найкращим рішенням цієї проблеми може бути навчання класифікатора на вибірках з приблизно рівним розподілів класів, але це в свою чергу може спричинити декілька нових проблем. По-перше, змінити частоту документів, які зустрічаються в реальних умовах, а значить погіршити роботу класифікатора. По-друге, створення вибірки стає ще складнішою задачею.

4.1.2 Точність (Precision)

Часто використовується разом з описаною нижче повнотою для оцінки якості алгоритмів. Визначається окремо для кожного класу. Її можна визначити, як відношення кількості правильно розпізнаних об’єктів даного класу до загальної кількості розпізнаних об’єктів даного класу. Формально зобразити точність можна наступним чином:

$$Pr = \frac{TP}{TP + FP},$$

де *TP* (true positive) – кількість правильно розпізнаних об’єктів даного класу, *FP* (false positive) – кількість помилкових розпізнавань об’єктів даного класу.

Наприклад, класифікатор розпізнав 15 текстів з образливим вмістом. При цьому 10 з них були відмічені, як образливі експертом, а 5 – нейтральні. В такому випадку точність класифікатора – 10 / (10 + 5) = 2 / 3.

Також важливо зазначити, що дана метрика може бути використана лише для бінарної класифікації.

4.1.3 Повнота (Recall)

Часто використовується разом з точність для оцінки якості алгоритмів. Також може бути використана для розрахунку інших похідних оцінок (F-міри та R-Precision). Аналогічно точності визначається окремо для кожного класу та може бути використана лише для бінарної класифікації. Повнота показує, яку долю об’єктів, які відносяться до даного класу було розпізнано правильно. Формальне зображення повноти:

$$Rec = \frac{TP}{TP + FN},$$

де *TP* – true positive – кількість правильно розпізнаних об’єктів даного класу, *FN* – false negative – кількість не розпізнаних об’єктів даного класу.

Наприклад, класифікатор перевірив 20 текстів, які містять образливий вміст і при цьому класифікував 15 з них правильно, а 5 відніс до нейтральних. То повнота даного класифікатора складає 15 / (15 + 5) = 0.75.

4.1.4 F-міра (F-score)

Зазвичай для створення класифікатора хорошої якості домагаються рівноваги повноти та точності. Саме для оцінки їх рівноваги

використовують F-міру. Оскільки нам потрібно отримати невелику різницю між усереднюваними даними – доцільно використати середнє гармонійне, яке при великих різницях між набором його чисел наближується до мінімального з них. Формально F-міра визначається так:

$$Fscore = \frac{2 \cdot Pr \cdot Rec}{Pr + Rec},$$

де Pr (precision) – точність, Rec (recall) – повнота.

В деяких задачах одна з метрик може бути важливіша за іншу. Наприклад, при визначенні спаму точність важливіша за повноту. Оскільки зайвий спам можна легко видалити, а якщо система автоматично позначить важливе повідомлення, як спам, то це може призвести до суттєвих втрат клієнта.

Саме для таких випадків використовують параметричну F-міру:

$$Fscore = \frac{(1 + a^2) \cdot Pr \cdot Rec}{a^2 \cdot Pr + Rec},$$

де Pr (precision) – точність, Rec (recall) – повнота, a – параметр з проміжку $[0; \infty]$. При $a = 0$, F-міра вироджується до точності, а при $a = \infty$ – до повноти.

В загальному параметрично F-міру можна охарактеризувати, як F-міру для випадку, коли повнота в a разів важливіша за точність.

4.2 Аналіз результатів

4.2.1 Перший розподіл набору

В даному тесті набір було розподіллено наступним чином – 66.6% набору було виділено для навчання, а інші – 33.3% для класифікації.

При використанні запропонованого методи детекції образливого вмісту було отримано наступну матрицю помилок:

$$\begin{matrix} 70 & 3 \\ 17 & 10 \end{matrix}$$

Виходячи з отриманої матриці можна розрахувати значення обраних нами метрик:

1. Частка правильно класифікованих текстів – 0.8.
2. Точність – 0.959.
3. Повнота – 0.805.
4. F-міра – 0.875.

Було отримано досить високу частку та точність та дещо гіршу повноту результатів класифікації. Це можна пояснити тим, що у вибірці є більшість текстів відносить до текстів, що містять образливий вміст (78%).

Оскільки в конкретно даному випадку досить важко сказати, що нам важливіше точність чи повнота, то складно було вирішено використовувати просту F-міру.

При детекції образливого вмісту за допомогою простого класифікатора Баєса було отримано наступну матрицю помилок:

$$\begin{matrix} 67 & 3 \\ 20 & 10 \end{matrix}$$

Виходячи з отриманої матриці можна розрахувати значення обраних нами метрик:

1. Частка правильно класифікованих текстів – 0.77.
2. Точність – 0.957.
3. Повнота – 0.77.
4. F-міра – 0.853.

В випадку простого класифікатора Баєса було отримано схожі за пропорціями результати з дещо меншими абсолютними значеннями.

Таблиця 1

Оцінка результатів для 66.6% на 33.3%

	Запропонований метод визначення образливого вмісту	Наївний Баєсів класифікатор
Частка	0.8	0.77
Точність	0.959	0.957
Повнота	0.805	0.77
F-міра	0.875	0.853
Час обробки, мс	67	10

Також варто звернути увагу, що запропонований метод показує суттєво гірший час, оскільки в ньому набагато складніший крок передобробки. Але на даний момент не було приділено увагу оптимізації цього кроку, тому швидкодію можна ще покращити. Крім того, в наш час можна вирішити це питання використавши мікросервісний підхід, якщо буде потрібно обробляти дуже великі масиви даних.

4.2.2 Другий розподіл набору

В даному тесті набір було розподіллено наступним чином – 90% набору було виділено для навчання, а інші – 10% для класифікації.

При використанні запропонованого методи детекції образливого вмісту було отримано наступну матрицю помилок:

$$\begin{matrix} 21 & 1 \\ 5 & 4 \end{matrix}$$

Виходячи з отриманої матриці можна розрахувати значення обраних нами метрик:

1. Частка правильно класифікованих текстів – 0.833.
2. Точність – 0.955.
3. Повнота – 0.808.
4. F-міра – 0.875.

Аналогічно результату з попереднього розподілу набору спостерігається досить висока частка правильно класифікованих текстів. Дещо вища ніж при 66% вибірки, адже ми використали більше даних для навчання. Та суттєве зміщення в бік точності.

При детекції образливого вмісту за допомогою простого класифікатора Баєса було отримано наступну матрицю помилок:

$$\begin{matrix} 20 & 0 \\ 6 & 4 \end{matrix}$$

Виходячи з отриманої матриці можна розрахувати значення обраних нами метрик:

1. Частка правильно класифікованих текстів – 0.8.
2. Точність – 1.
3. Повнота – 0.769.
4. F-міра – 0.869.

При використанні наївного басового класифікатора було отримано точність рівну 1, але це нівелюється тим, що інші метрики гірші. В тому числі і F-score. Також можна помітити, що кількісні метрики зросли з розміром набору для навчання.

Таблиця 2

Оцінка результатів для 90% на 10%

	Запропонований метод визначення образливого вмісту	Наївний Баєсів класифікатор
Частка	0.833	0.8
Точність	0.955	1
Повнота	0.808	0.769
F-міра	0.875	0.869
Час обробки, мс	26	8

4.3 Висновки

У даному розділі проведено аналіз розробленого методу автоматизованого визначення образливого вмісту текстових повідомлень в соціальних мережах. Метод дозволяє розв’язати задачу дослідження, даючи змогу створити автоматизовані системи з визначення образливого вмісту, які можуть бути використані в реальних умовах.

Проведено аналіз параметрів якості методів визначення образливого вмісту, що в даному випадку зводиться до бінарної класифікації. Виділено 4 основні метрики: частка правильно класифікованих текстів, точність, повнота та міра. Крім того було заміряно швидкодію алгоритму для поверхневого аналізу.

Розроблено програмне забезпечення призначене для аналізу запропонованого методи визначення образливого вмісту. В розробленому програмному забезпечені було імплементовано запропонований метод. Також для можливості порівняння якості роботи запропонованого методу з уже існуючими було реалізовано метод, що використовує простий наївний Баєсів класифікатор. Крім того, для навчання та тестування було відібрано та розмічено вибірку, що складається з 300 повідомлень (твітів) з соціальної мережі Twitter.

Завдяки використанню створеного програмного забезпечення було проведено аналіз показників якості визначення образливого вмісту. Запропонований метод показав кращі на 2%-5% результати точності визначення образливого вмісту, але суттєво гірший час. Проте при проведенні оптимізації кроку передобробки, використанні мікросервісного підходу або при використанні задач, що не вимагають швидкої обробки даних даний алгоритм має переваги над існуючими реалізаціями.

5 ПОБУДОВА БІЗНЕС-МОДЕЛІ

5.1 Опис проблеми

Автоматизована обробка природомовних текстів є галуззю комп'ютерної лінгвістики, що активно розвивається. Дана тенденція легко пояснюється стрімким зростанням об'ємів електронних текстових даних, які просто фізично стає неможливим обробити вручну. Відносно новим напрямком у автоматизованій обробці текстових даних є автоматизоване виявлення образливого вмісту.

Цей напрямок комп'ютерної лінгвістики є новим, але вже на даний момент дуже затребуваним та актуальним. З появою Інтернету та соціальних мереж, їх стрімкий розвиток та збільшення кількості користувачів виникла потреба у модерації повідомлень. Такі гіганти ринку як Facebook, Twitter, YouTube та подібні платформи впроваджують відповідні політики безпеки, щоб убезпечити користувачів від різноманітних ризиків. В рамках дотримання політики безпеки соціальні мережі модерують повідомлення, проте якість цієї модерації є низькою, що викликано наступними факторами [44].

По-перше, на сьогоднішній день єдиним методом модерації залишається використання найманої робочої сили, а саме – людей-модераторів. Саме ці працівники виконують моніторинг та модерацію образливих повідомлень, шляхом роботи у службах підтримки даних платформ. Даний метод має свій перелік недоліків.

Одним з таких недоліків є те, що процес модерації є досить повільним, через швидкість роботи людини-модератора. А зважаючи на те, що щохвилинно в мережі з'являється сотні тисяч нових постів чи коментарів, то перевірити весь контент просто неможливо.

Наступним недоліком є те, що утримання великої кількості модераторів є досить затратною опцією: до затрат відноситься заробітна платня модераторів, соціальні виплати, страховки (що в деяких країнах є

обов’язковим), утримання їх робочих місць, оренда приміщень та інші супутні витрати.

Також суттєвим недоліком є людський фактор у визначенні образливого вмісту, адже різні працівники можуть трактувати однаковий текст по різному.

По-друге, на сьогоднішній день кожен користувач може поскаржитися на пост або коментар іншого користувача. Якщо таких скарг буде декілька, то даний пост чи коментар буде автоматично заблоковано (і немає жодного значення, де було написано даний коментар, для якого кола осіб він був відкритим і чи взагалі містив образливий вміст). Дана опція несе в собі декілька вразливосте. Першою вразливістю є так звані замовні атаки, коли людям платять за скарги на чиїсь пости, наприклад політиків. Крім того, дані атаки можуть спричинятися не реальними людьми, а «фабриками ботів» (спеціально написаними програмами, що емулюють діяльність справжніх користувачів в соціальних мережах).

Все вище описане можна узагальнити у схемі «Дерева проблем» (рис. 5.1.1).

5.2 Зацікавлені сторони

У вирішенні описаної вище проблеми існує декілька зацікавлених сторін.

Найбільш очевидною та впливовою зацікавленою стороною, є самі соціальні мережі. Саме соціальні платформи, такі як Facebook, Twitter, YouTube та інші витрачають щорічно великі кошти на утримання регіональних служб підтримки, для зменшення кількості образливого вмісту.

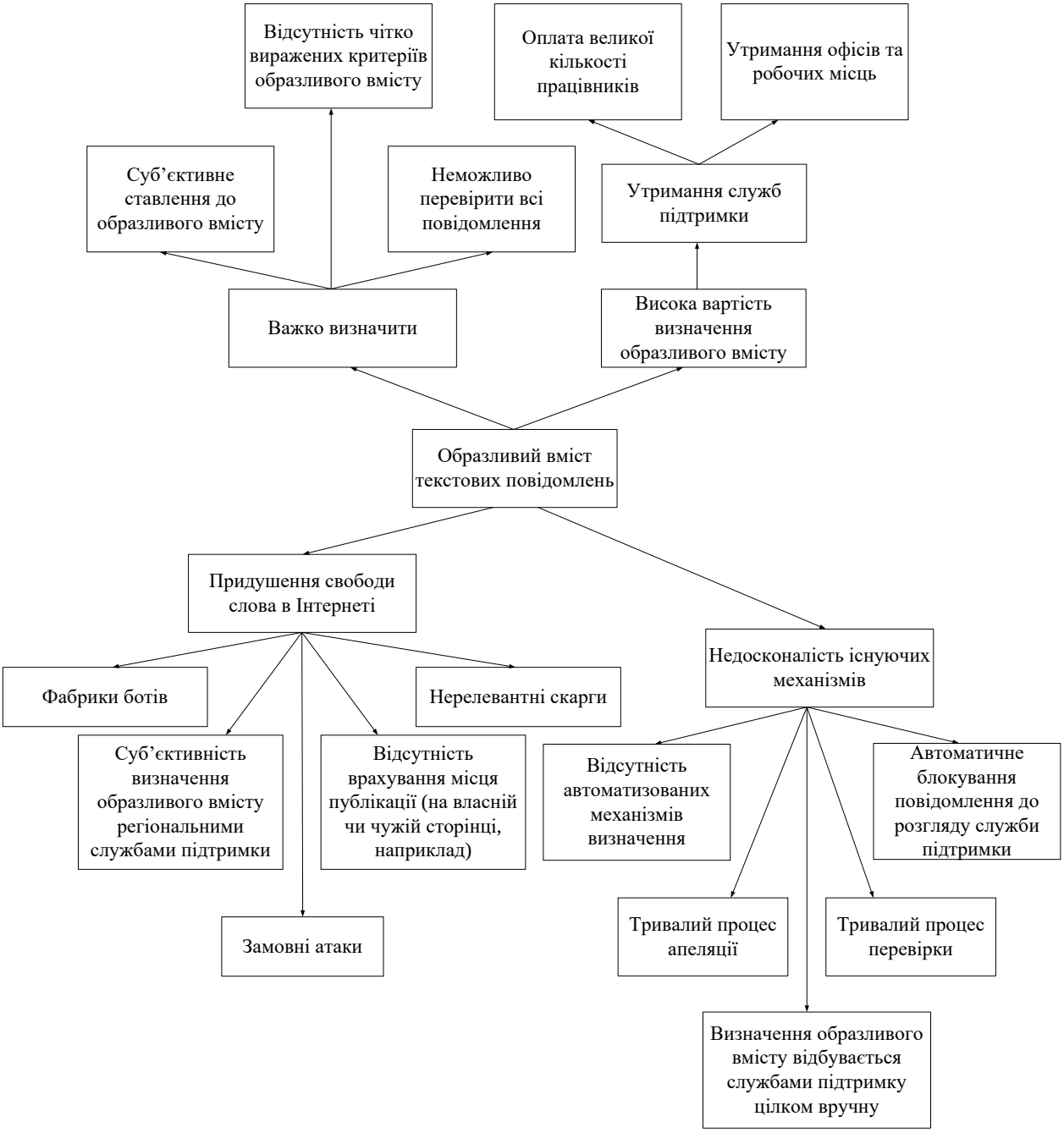

Рис. 5.1. Дерево проблем

Також зацікавленими сторонами є так звані «content makers» (контент мейкери або виробники контенту). Саме їх пости стають об'єктами замовних атак, а блокування чи статус «18+» несе значні втрати прибутків. Саме контент мейкери мають величезні аудиторії, а за їх допомогою соціальні платформи просувають рекламу.

Варто зазначити, що внесення до категорії «18+» автоматично блокує появу даного контенту в списку популярних повідомлень та зі

списку рекомендованих постів. Для рекламодавців та контен мейкерів це значить зниження переглядів, а відповідно і прибутків. В більш глибокій перспективі це може спричинити і зменшення доходів у самої соціальної мережі.

Якщо якась з платформ запропонує популярному контент мейкеру захист від атак та блокувань, то він може змінити платформу, на якій здійснює діяльність. Це спричинить відплив аудиторії на інший ресурс, а зважаючи на те, що топ-блогери та контент мейкери мають мільйонні, а подекуди і десяти мільйонні аудиторії, це спричинить зменшення рекламних колоборацій та відповідно прибутків для платформи, яку покинули. Тому вплив контент-мейкерів є також суттєвим.

Наступною категорією зацікавлених осіб є рекламодавці. Ця категорія зацікавлених осіб потребує впевненості в тому, що канал, публічна сторінка чи пост, на якій була замовлена реклама, буде побачена максимальною кількістю користувачів (іншими словами не буде заблокована, або віднесена до категорії «18+» без об'єктивних на те причин). Так як рекламодавці є основним ресурсом доходу соціальних мереж, то їх вплив на вирішення даної проблеми є теж високим.

І останньою по рахунку, але не по змісту, є категорія користувачів соціальним мереж. По-перше, саме користувачі забезпечують можливість впровадження реклами, а відповідно і заробітку соціальних мереж. По-друге, конкуренція на ринку соціальним мереж є досить високою, існує безліч соціальних мереж аналогів, тому соціальні мережі конкурують за користувача. Якщо улюблений блогер був заблокований, або доданий до списку «18+», то рекомендацій на нові пости користувач не буде одержувати і його зацікавленість соціальною мережею буде падати. Відносно попередніх зацікавлених осіб, вплив даної сторони є опосередкованим, але тим не менше він існує і ним не варто нехтувати.

У табл. 3 зведено усі групи зацікавлених сторін, їх інтереси, та вплив (міра зацікавленості у вирішенні наявних проблем).

Таблиця 3

Зацікавлені сторони

Зацікавлена сторона	**Інтерес зацікавленої особи**	**Вплив зацікавленої особи**	**Стратегії приваблення зацікавлених сторін**
Соціальні мережі	Економний та надійний спосіб виявлення образливого вмісту	Високий	Проведення презентації для представників зацікавлених осіб. Участь у спеціалізованих виставках та форумах.
Виробники контенту	Зниження ризику неправомірних блокувань та внесення до категорії «18+»	Середній	
Рекламодавці	Впевненість в охопленні рекламою широкої аудиторії (відсутність блокувань топ блогерів)	Середній	
Користувачі	Можливість отримати доступ до бажаного контенту	Низький	

5.3 Комерційне рішення. Основні характеристики

Відповідно до вище зазначених проблем, можна описати кінцевий продукт, що має їх вирішувати. Даний програмний продукт буде реалізовувати описаний у попередній розділах автоматизований метод визначення образливого вмісту, що базується на зв’язках користувачів. Даний метод дозволяє ефективно визначати образливий вміст текстових даних та вирішувати чи блокувати даний пост чи ні. Дані зміни не будуть помітними для користувачів чи контет мейкерів (за винятком зменшення необґрунтованих блокувань та віднесення до категорії «18+») або для

рекламодавців, проте для самих соціальних мереж впровадження даного програмного забезпечення буде помітним одразу.

По-перше, зменшиться кількість витрат на утримання служб підтримки по всьому світу. По-друге, соціальна мережа стане більш привабливо для нових контен мейкерів, користувачів та рекламодавців відповідно.

Саме тому, очевидно, що клієнтом даного продукту є самі соціальні мережі, а співпраця буде побудована на моделі співробітництва бізнесу для бізнесу, або як він ще називається B2B [45].

Даний програмний продукт повинен легко інтегруватися з існуючими механізмами роботи соціальної мережі. У випадку виявлення образливого вмісту, дане програмне забезпечення посяде місце людини аналога. Тобто, замість того, щоб текст посту чи коментаря направлявся модератору, даний текст буде приходити на обробку розробленому методу, а відповідний результат відсипатиметься так само, як би це зробила людина. Зважаючи на вище сказане, можна зробити висновок, що проблем з інтеграцією з існуючими системами виникати не повинно, або вони повинні швидко і легко вирішитися за допомогою невеликої кількості спеціалістів.

5.4 Конкуренті переваги рішення

Так як вже зазначалося, що дана сфера є досить молодою, то реалізацій автоматизованого підходу визначення образливого вмісту наразі не існує. Зараз існують окремі наукові дослідження, але жодне з них не є розробленим заради даних цілей визначення образливого вмісту у соціальних медіа чи соціальних мережах (існують невеличкі програми для виявлення грубих повідомлень в мережі Twitter, проте дана соціальна мережа обмежує свої повідомлення 160 символами, що не робить таке програмне забезпечення універсальним для всіх соціальних мереж). Тому

єдиним конкурентом залишаються служби підтримки з їх людськими ресурсами.

Основний конкурент має безліч недоліків, які були описані в пункті 5.1, Майже всі недоліки, що має аналог, вирішуються за допомогою запропонованого методу автоматизованого визначення образливого вмісту природомовних тестових даних.

Отже, конкурентними перевагами програмного продукту, що пропонується, є:

- зменшення вартості виявлення образливого вмісту в порівнянні з існуючими аналогами;
- збільшення швидкості обробки підозрілих на образливий вміст природомовних текстів;
- покращення якості виявлення образливого вмісту (відсутність суб'єктивізму та наявність уніфікованих критеріїв оцінки);
- унікальний метод визначення образливого вмісту, що враховує специфіку соціальних мереж.

5.5 Клієнти. Сегменти ринку споживання

Як вже зазначалося вище, клієнтами даного програмного забезпечення є соціальні мережі.

На сьогоднішній день великі соціальні мережі охоплюють десятки країн, що в свою чергу спричиняє сегментацію за мовною ознакою. Найбільш широковживаною мовою спілкування є англійська [46]. Саме англомовні країни одними з найбільш розвинених (наприклад, США, Канада, Велика Британія, Австралія), і відсоток населення, що користується соціальними мережами в них є досить високим [47]. Крім того, більшість штаб-квартир найбільш популярних у світі соціальних мереж знаходиться в Сполучених Штатах Америки. З вище сказаного, можна зробити висновок, що найбільш розвиненим є англомовний сегмент соціальних мереж.

Сегментація ринку за мовною ознакою, крім розміру ринку, спричинена інженерно-технічним фактором. Складність в реалізації даного методу іншими мовами пов’язана з тим, що додаткові методи такі, як передобробка тексту (стемінг та лематизація), найбільш якісно розроблені саме для англійської мови (це пов’язано з тим, що англійська мова є мовою міжнародного спілкування, а також з тим, що наукова спільнота розробляє більшість методів обробки природомовних текстів саме для англійської мови). І незважаючи на те, що описаний в роботі метод може бути застосований для будь-якої мови, проте для цього необхідні програмні реалізації передобробки тексту, що наприклад для української мови реалізувати досить важко через її лексичну та синтаксичну складність.

Також провести сегментацію ринку можна за обсягом користувачів, які обслуговує соціальна мережа: це великі соціальні мережі (такі як Facebook, Twitter, YouTube), середні (наприклад, tumblr) та малі (нішові соціальні мережі). Найчастіше необхідність виявленні образливого вмісту виникає на різноманітних платформах для обговорення, там, де зав’язується палка дискусія. В зв’язку з великою кількістю різноманітних користувачів, найбільш вразливою до виникнення образливого вмісту є великі соціальні мережі.

Крім того, соціальні мережі можна поділити за наступними ознаками: соціальні мережі для спілкування (Relationship networks), соціальні мережі для обміну медіа-контентом (Media sharing networks), соціальні мережі для відгуків і оглядів (Online reviews), соціальні мережі для колективних обговорень (Discussion forums), соціальні мережі для авторських записів (Social publishing platforms).

Перша категорія соціальних мереж є найпоширенішою і найбільш затребуваною соціальні медіа на сьогоднішній день. До них відносяться Facebook, Вконтакте, Однокласники, Linkedin. І хоча даний вид соціальних медіа не з'явився першим, він став визначальним для всієї галузі. Даний формат соціальних медіа одним з перших запропонував користувачам

створити безкоштовний персональний міні-сайт, який пізніше став відомий як профіль. Мережі взаємин намагаються запропонувати користувачам максимум можливостей в межах однієї платформи.

Умовно можна розділити мережі взаємин на наступні категорії:

- мережі персональних контактів;
- професійні мережі;
- дейтинг (сайти знайомств).

Даний вид соціальних медіа становить найбільший інтерес для бізнесу. Сьогодні сторінка бренду в Twitter або Facebook – це загальноприйнятий стандарт. Саме тому дані соціальні мережі є найбільш зацікавленими у пошуку альтернативних способів модерації повідомлень і можуть у перспективі стати споживачами запропонованого продукту.

Другий вид соціальних медіа, а саме соціальні мережі для обміну медіа-контентом, дає користувачам широкі можливості для обміну відео- та фото-контентом. Сюди відносяться Flickr, Instagram, YouTube, Vimeo, Vine, Snapchat. Відмінною особливістю є також і масштабування контенту: деякі пропонують публікувати короткі відеоролики, інші дають вам можливість створити власний відео канал. Проте, до будь якого виду контенту можна додати опис чи назву, або коментарі, які міститимуть образливий контент, тому дані мережі теж можуть використовувати запропонований методу для виявлення образливого вмісту.

Третій вид соціальних мереж є менш цікавим у контексті виявлення образливого вмісту. В основі зарубіжних мереж Yelp і Urbanspoon лежить геолокації і можливість залишати коментарі та рекомендації про локальному бізнесі. Airbnb і Uber, фокусуються на відгуках про місця проживання для мандрівників і приватних перевізників. Такі соціальні медіа – це величезна база знань, яка допомагає користувачам зібрати всю необхідну інформацію для прийняття рішення про покупку. Дані соціальні медіа більш зацікавленні у програмному забезпеченні з визначення тональності відгуків, а не образливого вмісту.

Соціальні мережі для колективних обговорень є місцями, де образливий вміст може з’являтися досить часто. Спільноти, форуми, Q & A-сервіси - одні з перших видів соціальних медіа. До сучасних представників даного виду можна віднести Quora, Reddit і Digg. В основі механіки взаємодія між користувачами лежить потреба в обміні знаннями. Даний вид соціальних мереж не є прихильником агресивної реклами. Бізнес менш зацікавлений в таких соціальних мережах (наприклад, рекламодавці), тому автоматизоване виявлення образливого вмісту для цього виду соціальних мереж є менш актуальним.

До соціальних мереж для авторських записів відносяться послуги для блогінгу і мікро-блогінгу, де користувачі створюють і публікують текстово-медійний контент. Сюди відносяться такі популярні платформи, як Twitter, Medium і Tumblr. Дані соціальні мережі, окрім Twitter, є досить нішовими, а деякі з них взагалі не мають обмежень для контенту, наприклад Tumblr, тому виявлення образливого вмісту для цього виду соціальних мереж є менш актуальним.

Отже, для співпраці, найбільш перспективними є великі соціальні мережі, що мають англомовний сегмент та відносяться до типу соціальних мереж для спілкування та соціальних мереж для обміну медіа-контентом, бо саме ці види соціальних мереж є найбільш привабливими для бізнесу та рекламодавців.

5.6 Унікальна ціннісна пропозиція

Ціннісна пропозиція – це пояснення того, як продукт вирішує проблему. Його можна скласти за формулою:

Ціннісне пропозицію = Проблема + Рішення / Продукт.

У дереві проблем було виділено низку проблем, а у зацікавлених сторонах – було визначено очікування відповідних сторін від продукту. Соціальні мережі бажають отримати економний та надійний спосіб виявлення образливого вмісту, виробники контенту – зниження ризику

неправомірних блокування та внесення до категорії «18+», рекламодавці хочуть впевненість в охопленні рекламою широкої аудиторії (відсутність блокувань топ блогерів), а користувачі – отримати можливість вільного доступу до бажаного контенту. Дійсно, запропоноване рішення, дозволяє частково задовольнити всі з наведених вище вимог зацікавлених сторін та вирішити наведені нижче проблеми.

Отже, основною унікальною ціннісною пропозицією є розроблений метод автоматизованого визначення образливого вмісту, що враховує зв'язки між користувачами, а основною перевагою є врахування особливостей соціальних мереж, які раніше ніким не було реалізовано чи запропоновано. Або іншими словами: «Relation based automatic method of abusive language detection».

5.7 Доходи та витрати

Сумарний дохід складається як сума доходу на продаж ліцензії на використання програмного забезпечення та надання послуг на його підтримку.

Продаж програмного забезпечення планується шляхом продажу ліцензії на програмне забезпечення, тобто укладання угоди, яка надає право використовувати програмне забезпечення. Вид ліцензії, який планується продаватися, це commercial (комерційна) ліцензія. Дана ліцензія передбачає використання закритого, власницького або пропрієтарного ПЗ. Пропрієтарне програмне забезпечення [48], (від англ. proprietary software) – це програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права [49]. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж [50]. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень.

Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію. Таке програмне забезпечення називається закритим. Проте, код власницького продукту може бути й відкритим, але власник може обмежити права користувача умовами ліцензії.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути й безплатне (тобто, некомерційне) програмне забезпечення.

Саме тому ліцензія буде розповсюджуватися на комерційне програмне забезпечення.

Платна технічна підтримка включає в себе реакцію на запит споживача та усунення недоліків роботи програми в строки погоджені договором використання програмного забезпечення.

Таблиця 4

Витрати на реалізацію проекту

Найменування витрат	**1-й місяць, т. $**	**2-й місяць, т. $**	**3-й місяць, т. $**	**4-й місяць, т. $**	**5-й місяць, т. $**	**6-й місяць, т. $**
Загальні витрати	2	2	2	2	2	2
ЗП		25	25	25	25	25
Витрати	**2**	**27**	**27**	**27**	**27**	**27**
Заплановані прибутки						
Результат(без оподаткування)	**-2**	**-27**	**-27**	**-27**	**-27**	**-27**

Таблиця 4 продовження

Найменування витрат	**7-й місяць, т. $**	**8-й місяць, т. $**	**9-й місяць, т. $**	**10-й місяць, т. $**	**11-й місяць, т. $**	**12-й місяць, т. $**	**Загальні результати, т. $**
Загальні витрати	2	2	2	2	2	2	24
ЗП	28	28	28	28	28	28	293
Витрати	**30**	**30**	**30**	**30**	**30**	**30**	**317**
Заплановані прибутки	**75**	**75**	**75**	**75**	**75**	**75**	**450**
Результат (без оподаткування):	**45**	**45**	**45**	**45**	**45**	**45**	**133**

До витрат відноситься:

- утримання персоналу для надання технічної підтримки (виплати заробітних плат, соціальних виплат);
- утримання робочих місць для персоналу (оплата за оренду офісу та комунальні послуги);
- податкові витрати;

- оплата послуг юриста, бухгалтера, прибиральниці.

Детальніше ознайомитися з витратами на реалізацію проекту та прогнозованим прибутками можна з таблиці 4.

5.8 Бізнес-модель

Узагальнимо, все написане вище у лаконічну бізнес-модель у вигляді lean canvas.

Споживачі: великі соціальні мережі типу соціальних мереж для спілкування та соціальних мереж для обміну медіа-контентом.

Проблема: відсутність автоматизованих рішень; висока вартість людських послуг; низька якість послуг модераторів; людський фактор та суб'єктивізм; відсутність уніфікованих критеріїв образливого вмісту.

Рішення: програмне забезпечення, що визначає образливий вміст шляхом використання автоматизованого методу визначення образливого вмісту природомовних текстів, що заснований на зв'язках користувачів.

Унікальна ціннісна пропозиція: програмне забезпечення, що реалізує новий метод визначення образливого вмісту: «Relation based automatic method of abusive language detection»; зменшення витрат соціальних мереж на служби підтримки.

Потоки доходів: доходи від продажу ліцензій; доходи від підтримки програмного забезпечення;

Структура витрат: утримання персоналу для надання технічної підтримки (виплати заробітних плат, соціальних виплат); утримання робочих місць для персоналу (оплата за оренду офісу та комунальні послуги); податкові витрати; оплата послуг юриста, бухгалтера, прибиральниці.

Таблиця 5

Канва бізнес-моделі

<table>
<tr>
<td rowspan="2">Проблема

відсутність автоматизованих рішень;

висока вартість людських послуг;

низька якість послуг модераторів;

людський фактор та суб’єктивізм;

відсутність уніфікованих критеріїв образливого вмісту</td>
<td>Рішення

програмне забезпечення, що визначає образливий вміст шляхом використання автоматизованого методу визначення образливого вмісту природомовних текстів, що заснований на зв’язках користувачів</td>
<td rowspan="2">Унікальна ціннісна пропозиція

«Relation based automatic method of abusive language detection»;

зменшення витрат соціальних мереж на служби підтримки</td>
<td>Прихована перевага

врахування специфіки соціальних мереж при автоматизованому визначенні образливого вмісту</td>
<td rowspan="2">Споживачі

великі соціальні мережі типу соціальних мереж для спілкування та соціальних мереж для обміну медіа-контентом</td>
</tr>
<tr>
<td>Ключові метрики

кількість проданих ліцензій</td>
<td>Канали

через відділи співпраці/інтеграції відповідних соціальних мереж</td>
</tr>
<tr>
<td colspan="3">Структура витрат

утримання персоналу для надання технічної підтримки (виплати заробітних плат, соціальних виплат);
утримання робочих місць для персоналу (оплата за оренду офісу та комунальні послуги);
податкові витрати;
оплата послуг юриста, бухгалтера, прибиральниці</td>
<td colspan="2">Потоки доходів

доходи від продажу ліцензій;
доходи від підтримки програмного забезпечення</td>
</tr>
</table>

Також в канву бізнес-моделі включаються структурні блоки: прихована перевага (перевага, яку не можливо скопіювати або купити), ключові метрики (основні показники, що вимірюються) та канали (шляхи до користувачів).

Канали: через відділи співпраці/інтеграції відповідних соціальних мереж.

Ключові метрики: кількість проданих ліцензій.

Прихована перевага: врахування специфіки соціальних мереж при автоматизованому визначенні образливого вмісту.

Бізнес-модуль наведена у зведеному вигляді у таблиці 5.

Отже, зважаючи на дані у таблиці 5, можна зробити висновок, що запропонований проект, який реалізує описаний у дисертацій метод визначення образливого вмісту, має перспективи у своїй подальшій реалізації. Звичайно, проведений аналіз не враховує всіх ризиків та факторів, таких як специфіка оподаткування у країні ведення бізнесу, проте навіть наявних досліджень достатньо, щоб прогнозувати комерційний успіх продукту та його окупаємість.

5.9 Висновки

У даному розділі було проведено аналіз поточної ситуації у сфері визначення образливого вмісту, виявлено наявні проблеми та підсумовано їх у відповідному дереві проблем. Наряду з проблемами було виділено основні зацікавлені сторони у вирішенні існуючих недоліків, ступінь впливу даних сторін на вирішення проблем. Як наслідок було запропоновано комерційне рішення з конкурентними перевагами, що задовольняє інтереси зацікавлених осіб, та виділено унікальну ціннісну пропозицію запропонованого продукту. Було проведено аналіз майбутніх клієнтів, досліджено сегменти ринку споживання. Це дозволило спрогнозувати потенційні доходи та витрати на реалізацію продукту. У результаті була описана бізнес-модель, що обґрунтовує доцільність реалізації даного продукту та прогнозує його потенційну окупаємість та прибутковість в подальшому.

ВИСНОВКИ

У даній роботі виконані наступні завдання:

1. Проаналізовано існуючі підходи та методи автоматизованого визначення образливого вмісту, визначено їхні основні переваги та недоліки.

2. проведено вивчення специфіки публікацій Інтернет-користувачів у соціальних мережах та сформовано перелік їх характеристик, які можуть бути враховані під час визначення образливого вмісту, а саме: наявність символів, що можуть призвести до заплутування сприйняття слів і фраз; контекст (на рівні повідомлень в соціальних мережах); контекст (на рівні зв’язків користувачів в соціальних мережах); наявність емодзі.

3. Проведено аналіз процесу детекції образливого вмісту текстових даних з точки зору можливості урахування в ньому специфіки повідомлень користувачів соціальних мереж, а саме запропоновано модифікацію етапів передоброблення тексту та аналізу результатів класифікації.

4. Розроблено метод визначення образливого вмісту повідомлень Інтернет-користувачів в соціальних мережах на основі комплексного урахування характеристик повідомлень в соціальних мережах та специфіки стосунків між різними групами відвідувачів соціальних мереж.

5. Запроваджено специфічний формат вхідних даних, що дозволяє реалізувати запропонований підхід до аналізу результатів класифікації методу визначення образливого вмісту текстових повідомлень в соціальних мережах, та який передбачає відображення зв’язків між користувачами та типів повідомлень в соціальних мережах

6. Реалізоване програмне забезпечення для визначення образливого вмісту повідомлень Інтернет-користувачів, яке реалізує запропонований метод.

7. Проведено тестування розробленої системи на відібраній та розміченій вибірці, що складається з 300 повідомлень (твітів) з соціальної мережі Twitter.

Розробка системи автоматизованого визначення образливого вмісту, що базується на запропонованому методі, показала покращені результати якості в порівнянні методом без відповідних модифікацій етапів передобробки та аналізу результатів.

Перспективний напрямком подальших досліджень автор вважає оптимізацію алгоритму інтерпретації емодзі символів у відповідні слова та фрази, щоб зменшити часові показники виконання запропонованого методу.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Internet World Stats [Електронний ресурс] — Режим доступу: https://www.internetworldstats.com/stats.htm (дата звернення 30.03.2018). — Назва з екрана.

2. Cyber Bullying Law and Legal Definition [Електронний ресурс]. — Режим доступу: https://definitions.uslegal.com/c/cyber-bullying/ — (дата звернеття 16.02.2018) — Назва з екрана.

3. *Заболотня Т.М., Бартков'як А.Ю*. Програмна бібліотека методів аналізу тональності відгуків Інтернет-користувачів // "Informatics and Computer Technics Problems". Тези доповідей. – Чернівці. 21-24 травня 2016. – С. 85-87.

4. *Ying Ch., Yilu Zh., Sencun Zh., Heng X.* Detecting Offensive Language in Social Media to Protect Adolescent Online Safety / *Ch. Ying, Zh. Yilu, Zh. Sencun, X. Heng* // — SOCIALCOM-PASSAT '12 Proceedings of the 2012 ASE/IEEE International Conference on Social Computing, 2012 — 2012 — Pp. 71-81.

5. Stanford NLP [Електронний ресурс] — Режим доступу: https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html (дата звернення 30.01.2018). — Назва з екрана.

6. Stanford NLP [Електронний ресурс] — Режим доступу: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html (дата звернення 30.01.2018). — Назва з екрана.

7. *Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y.* and *Chang, Y.*, Abusive Language Detection in Online User Content, / *C. Nobata, J Tetreault, A. Thomas, Y. Mehdad* and *Y. Chang* // — 25th International Conference on World Wide Web Pages: Montréal, Québec, Canada, 2016 — 2016 — Pp. 145-153.

8. Удовиченко Г. М. Загальне мовознавство. Історія лінґвістичних учень. — К. : Вища школа, 1980. — С. 5.

9. Кочерган М. П. Вступ до мовознавства. — К. : Видавничий центр «Академія», 2005. — С. 7.

10. Palmer, F. R., Mood and Modality. — Cambridge University Presents, 2001. — P. 33.

11. *E. Pitler* and *A. Nenkova* Using syntax to disambiguate explicit discourse connectives in text / *Pitler E.* and *Nenkova A.* // — The ACL-IJCNLP 2009 Conference Short Papers, Suntec, Singapore, 2009. — 2009 — Pp. 13-16.

12. Carol Lynn, Moder, Aida Martinovic-Zic Discourse Across Languages and Cultures. — John Benjamins Publishing Company, 2004. — P. 117.

13. *A. Broder, S. Glassman, M. Manasse,* and *G. Zweig* Syntactic clustering of the web / *Broder A., Glassman S., Manasse M.,* and *Zweig G.* // — Computer Networks and ISDN Systems. 29 (8), 1997. — 1997 — Pp. 1157-1166.

14. *D. Jurafsky,* and *J.H. Martin* Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. / *Jurafsky, D.* and *Martin, J.H.* // — Pearson Prentice Hall, 2009. — 2009 — P. 988.

15. B. Liu. Sentiment Analysis and Opinion Mining. — Morgan Claypool Publishers, 2012.

16. *M. Surdeanu, M. Ciaramita,* and *H. Zaragoza* Learning to rank answers to non-factoid questions from web collections. / *Surdeanu M., Ciaramita M.,* and *Zaragoza H.* // — Computational Linguistics, vol. 37, 2011. — 2011 — Pp. 351–383.

17. T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. CoRR, abs/1301.3781, 2013.

18. *Q. Le* and *T. Mikolov* Distributed representations of sentences and documents. / *Le Q.* and *Mikolov T.* // — Proceedings of the 31st International Conference on Machine Learning (ICML-14), 2014. — 2014 — Pp. 1188–1196.

19. *N. Djuric, H. Wu, V. Radosavljevic, M. Grbovic,* and *N. Bhamidipati* Hierarchical neural language models for joint representation of streaming documents and their content. / *Djuric N., Wu H., Radosavljevic V., Grbovic M.,* and *Bhamidipati N.* // — In International World Wide Web Conference (WWW), 2015. — 2015.

20. *N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic,* and *N. Bhamidipati* Hate speech detection with comment embeddings. / *Djuric N., Zhou J., Morris R., Grbovic M., Radosavljevic V.,* and *Bhamidipati N.* // — In International World Wide Web Conference (WWW), 2015. — 2015.

21. Вікіпедія вільна енциклопедія [Електронний ресурс] — Режим доступа: https://uk.wikipedia.org/wiki/Соціальна_мережа (дата звернення 30.03.2018). — Назва з екрана.

22. Навчальні матеріали онлайн [Електронний ресурс] — Режим доступа: http://pidruchniki.com/18421120/ritorika/pobutove_spilkuvannya (дата звернення 30.03.2018). — Назва з екрана.

23. *Tauch C. and Kanjo E.,* The roles of emojis in mobile phone notifications, / *C.Tauch, E. Kanjo* // — International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp Adjunct 2016 — 2016 — Pp. 1560–1565.

24. *Miller H., Thebault-Spieker J., Chang S., Johnson I. L., Terveen L. G.,* and *Hecht B.*, "blissfully happy" or "ready to fight": Varying interpretations of emoji, / *H. Miller, J. Thebault-Spieker, S. Chang, I. L. Johnson, L. G. Terveen, B. Hecht* // — 10th International Conference on Web and Social Media, ICWSM 2016 — 2016 — Pp. 259–268.

25. *Lu X., Ai W., Liu X., Li Q., Wang N., Huang G.* and *Mei Q.*, Learning from the ubiquitous language: An empirical analysis of emoji usage of smartphone users, / *X. Lu, W. Ai, X. Liu, Q. Li, N. Wang, G. Huang, Q. Mei* // — Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp 2016 — 2016 — Pp. 770–780.

26. *Заболотня Т.М., Соколовська А.В.* Підхід до визначення образливого вмісту текстових повідомлень в соціальних мережах // ”Прикладна математика та комп’ютинг. ПМК 2018”. Тези доповідей. – Київ. 21-23 березня 2018. – С. 243-249.

27. Poynter [Електронний ресурс] – Режим доступу: http://www.poynter.org/2013/25-of-people-have-posted-anonymous-comments-pew-finds/222912/ – (09.10.2016).

28. *Ashley A. Anderson, Dominique Brossard, Dietram A. Scheufele, Michael A. Xenos, Peter Ladwig* The “Nasty Effect:” Online Incivility and Risk Perceptions of Emerging Technologies / *Anderson Ashley A., Brossard Dominique, Scheufele Dietram A., Xenos Michael A., Ladwig Peter* // Journal of Computer-Mediated Communication, Volume 19, Issue 3. — 2014. — Pp. 373-387.

29. TIOBE Index for May 2018 [Електронний ресурс] – Режим доступу: https://www.tiobe.com/tiobe-index/ – (09.10.2017).

30. Особенности языка C# [Електронний ресурс]. — Режим доступу : http://phpbb3.ru/?p=7831. — (09.02.2016).

31. Microsoft [Електронний ресурс]. — Режим доступу : https://www.microsoft.com/net/download/windows. — (09.02.2018).

32. Introducing .NET Core [Електронний ресурс]. — Режим доступу : https://blogs.msdn.microsoft.com/dotnet/2014/12/04/introducing-net-core/. — (09.05.2018).

33. .NET 2015 Overview [Електронний ресурс]. — Режим доступу : https://channel9.msdn.com/Events/Visual-Studio/Connect-event-2015/NET-2015-Overview. — (09.05.2018).

34. .NET Core - .NET становится кросс-платформенной [Електронний ресурс]. — Режим доступу : https://msdn.microsoft.com/magazine/mt694084. — (09.05.2018).

35. Платформа .Net та її застосування для ООП [Електронний ресурс]. — Режим доступу : http://www.znannya.org/?view=csharp-dotNET. — (09.02.2016).

36. Visual C++ Team Blog [Електронний ресурс]. — Режим доступу : https://blogs.msdn.microsoft.com/vcblog/2013/07/19/c99-library-support-in-visual-studio-2013/. — (09.05.2018).

37. F# at Microsoft Research [Електронний ресурс]. — Режим доступу : https://www.microsoft.com/en-us/research/project/f-at-microsoft-research/?from=http%3A%2F%2Fresearch.microsoft.com%2Fen-us%2Fum%2Fcambridge%2Fprojects%2Ffsharp%2F. — (09.05.2018).

38. Best python ide for python programming [Електронний ресурс]. — Режим доступу: http://www.pythonicquest.com/article/best-python-ide/. — (09.05.2018).

39. Announcing Visual Studio 2017 General Availability… and more [Електронний ресурс]. — Режим доступу: https://blogs.msdn.microsoft.com/visualstudio/2017/03/07/announcing-visual-studio-2017-general-availability-and-more/. — (09.05.2018).

40. Ethnologue [Електронний ресурс] — Режим доступу: http://www.ethnologue.com/statistics/size — (04.06.2016).

41. Eric Freeman Head First Design Patterns / Freeman Eric, Bates Bert, Sierra Kathy, Robson Elisabeth // O'Reilly Media; 1st edition (October 2004). — 2004. — P. 694.

42. Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship. — Pearson Education, 2008. — P. 157. — ISBN 978-0-13-608325-2.

43. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley. pp. 97ff. ISBN 0-201-63361-2.

44. «A brief history of Natural Language Processing: [The University of Birmingham]». — Режим доступу: http://www.cs.bham.ac.uk/~pjh/sem1a5/pt1/pt1_history.html. — (13.10.2017).

45. «Cambridge Dictionary: [Cambridge University Press] ». — Режим доступу: https://dictionary.cambridge.org/dictionary/english/business-to-business. — (12.10.2017).

46. «Summary by language size: [Ethnologue]». – Режим доступу: https://www.ethnologue.com/statistics/size. — (12.10.2017).

47. «Global social media research summary 2017: [Smart Insights] ». – Режим доступу: https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/. — (19.12.2017).

48. Тлумачний словник з інформатики / Г.Г. Півняк, Б.С. Бусигін, М.М. Дівізінюк та ін. – Д., Нац. гірнич. ун-т, 2010. – С. 495][Міжнародний союз електрозв’язку; переклад – Юрій Пероганич (2006-11-23). Словник термінів. Всесвітній саміт з питань інформаційного суспільства. Підсумкові документи. с. 84. Архів оригіналу за 2013-06-25

49. Плескач В. Л. (2008). Інтелектуальна власність як джерело капіталу підприємств (PDF). Наукові праці НДФІ. Збірник наукових праць. Випуск 2 (43) (2008). НДФІ; Національна бібліотека України імені В.І. Вернадського. с. 142. Архів оригіналу за 2013-06-25

50. Проект Закону про використання Відкритих і Вільних форм інтелектуальної власності, Відкритих форматів даних та Відкритого (Вільного) програмного забезпечення в державних установах і державному секторі економіки. Ukrainian Association of developers and users of Free and Open Source Software. 2005-12-01. Стаття 1. пп.8, 26. Архів оригіналу за 2013-06-25.

ДОДАТКИ

Додаток 1

Лістинги

```
namespace AbusiveLanguageDetection.DataTypes
{
    using System.Collections.Generic;

    public class PostData
    {
        private User _author;
        private User _complainer;
        private PublicationType _publicationType;
        private List<User> _markedPeople;
        private string _text;

        public User Author
        {
            get { return _author; }
            private set { _author = value; }
        }

        public User Complainer
        {
            get { return  _complainer; }
            private set { _complainer = value; }
        }

        public PublicationType PublicationType
        {
            get { return _publicationType; }
            private set { _publicationType = value; }
        }

        public List<User> MarkedPeople
        {
            get { return  _markedPeople; }
            set { _markedPeople = value; }
        }

        public string Text
        {
            get { return  _text; }
            set { _text = value; }
        }

        public PostData(User author, User complainer, PublicationType
publicationType, string text)
        {
            Author = author;
            Complainer = complainer;
            PublicationType = publicationType;
            Text = text;
        }

        public PostData(User author, User complainer, PublicationType
publicationType, List<User> markedPeople,
            string text)
            : this(author, complainer, publicationType, text)
        {
            MarkedPeople = markedPeople;
        }

        public bool IsComplainerMarked()
        {
            return _markedPeople != null &&
_markedPeople.Contains(_complainer);
        }
```

```
        public bool IsPrivatePublication()
        {
            return _publicationType == PublicationType.PrivatePersonalPage;
        }

        public bool IsPersonalPublicPublication()
        {
            return _publicationType == PublicationType.PublicPersonalPage;
        }

        public bool IsPublicOnOtherPage()
        {
            return _publicationType == PublicationType.AnotherUserPage ||
                   _publicationType == PublicationType.CommunityPage;
        }
    }
}
```

```
namespace AbusiveLanguageDetection.DataTypes
{
    public enum PublicationType
    {
        PrivatePersonalPage,
        PublicPersonalPage,
        AnotherUserPage,
        CommunityPage
    }
}
```

```
using System;

namespace AbusiveLanguageDetection.DataTypes
{
    public class User : IEquatable<User>
    {
        private string _id;

        private string _name;
        private string _surname;

        public string Id
        {
            get { return _id; }
            private set { _id = value; }
        }

        public string Name
        {
            get { return _name; }
            private set { _name = value; }
        }

        public string Surname
        {
            get { return _surname; }
            private set { _surname = value; }
        }

        public User(string id, string name, string surname)
```

```
        {
            Id = id;
            Name = name;
            Surname = surname;
        }

        public bool Equals(User other)
        {
            if (ReferenceEquals(null, other)) return false;
            if (ReferenceEquals(this, other)) return true;
            return string.Equals(_id, other._id)
                   && string.Equals(_name, other._name)
                   && string.Equals(_surname, other._surname);
        }

        public override bool Equals(object obj)
        {
            if (ReferenceEquals(null, obj)) return false;
            if (ReferenceEquals(this, obj)) return true;
            return obj.GetType() == GetType() && Equals((User) obj);
        }

        public override int GetHashCode()
        {
            unchecked
            {
                int hashCode = _id?.GetHashCode() ?? 0;
                hashCode = (hashCode * 397) ^ (_name?.GetHashCode() ?? 0);
                hashCode = (hashCode * 397) ^ (_surname?.GetHashCode() ??
0);
                return hashCode;
            }
        }
    }
}
```

```
using System.IO;
using AbusiveLanguageDetection.TextProcessing.Classifier.NaiveBayes;

namespace AbusiveLanguageDetection.Detector.Creation
{
    public class BayesDetectorBuilder : Builder
    {
        public override void BuildClassifier()
        {

            string neutralEvidencePath =
Path.Combine(Configuration.BaseFolder, Configuration.NeutralClass);
            IEvidence neutralEvidence = new
Evidence(Configuration.NeutralClass, neutralEvidencePath);

            string abusiveEvidencePath =
Path.Combine(Configuration.BaseFolder, Configuration.AbusiveClass);
            IEvidence abusiveEvidence = new
Evidence(Configuration.AbusiveClass, abusiveEvidencePath);

            Classifier = new NaiveBayesClassifier(neutralEvidence,
abusiveEvidence);
        }
    }
}
```

```
using System.IO;
using AbusiveLanguageDetection.TextProcessing.Classifier;
using AbusiveLanguageDetection.TextProcessing.Emoji;
using AbusiveLanguageDetection.TextProcessing.SpecialSymbol;

namespace AbusiveLanguageDetection.Detector.Creation
{
    public abstract class Builder
    {
        protected IEmojiHandler EmojiHandler;
        protected ISpecialSymbolHandler SpecialSymbolHandler;
        protected IClassifier Classifier;

        public virtual void BuildEmojiHandler()
        {
            EmojiHandler = new EmojiHandler();
        }

        public virtual void BuildSpecialSymbolHandler()
        {
            string specialSymbolsFile =
Path.Combine(Configuration.BaseFolder, Configuration.SpecialSymbols);
            SpecialSymbolHandler = new
SpecialSymbolHandler(specialSymbolsFile);
        }

        public abstract void BuildClassifier();

        public virtual IAbusiveLanguageDetector GetDetector()
        {
            return new AbusiveLanguageDetector(Classifier,
SpecialSymbolHandler, EmojiHandler);
        }

    }
}


namespace AbusiveLanguageDetection.Detector.Creation
{
    public class Director
    {
        public void Construct(Builder builder)
        {
            builder.BuildEmojiHandler();
            builder.BuildSpecialSymbolHandler();
            builder.BuildClassifier();
        }
    }
}



using System.Linq;
using System.Collections.Generic;
using AbusiveLanguageDetection.TextProcessing;
using AbusiveLanguageDetection.DataTypes;
using AbusiveLanguageDetection.TextProcessing.Classifier;
using AbusiveLanguageDetection.TextProcessing.Classifier.NaiveBayes;
using AbusiveLanguageDetection.TextProcessing.Emoji;
using AbusiveLanguageDetection.TextProcessing.SpecialSymbol;
```

```
namespace AbusiveLanguageDetection.Detector
{
    public class AbusiveLanguageDetector : IAbusiveLanguageDetector
    {
        private readonly IClassifier _classifier;
        private readonly ISpecialSymbolHandler _specialSymbolHandler;
        private readonly IEmojiHandler _emojiHandler;

        internal AbusiveLanguageDetector(IClassifier classifier,
ISpecialSymbolHandler specialSymbolHandler, IEmojiHandler emojiHandler)
        {
            _classifier = classifier;
            _specialSymbolHandler = specialSymbolHandler;
            _emojiHandler = emojiHandler;
        }

        public bool IsAbusiveLanguage(PostData postData)
        {
            List<string> content = PreProcessing(postData.Text);

            IDictionary<string, double> classificationResult =
_classifier.Classify(content, ClassifierHelper.ExcludeList);

            return MakeDecision(postData,
classificationResult[Configuration.AbusiveClass]);
        }

        private List<string> PreProcessing(string data)
        {
            if (Configuration.UseEmojiHandler)
            {
                data = HandleEmoji(data);
            }

            if (Configuration.UseSpecialSymbolHandler)
            {
                data = HandleSpecialSymbols(data);
            }

            List<string> content = Tokenizer.TokenizeNow(data).ToList();

            return content;
        }

        private string HandleEmoji(string content)
        {
            return _emojiHandler.HandleEmoji(content);
        }

        private string HandleSpecialSymbols(string content)
        {
            return _specialSymbolHandler.HandleSpecialSymbol(content);
        }

        private bool MakeDecision(PostData postData, double probability)
        {
            if (postData.IsPrivatePublication())
            {
                return false;
            }

            if (postData.IsPersonalPublicPublication() && probability >
0.7)
            {
```

```
                return true;
            }

            if ((postData.IsComplainerMarked() ||
postData.IsPublicOnOtherPage()) && probability > 0.5)
            {
                return true;
            }

            return false;
        }
    }
}


using AbusiveLanguageDetection.DataTypes;

namespace AbusiveLanguageDetection.Detector
{
    public interface IAbusiveLanguageDetector
    {
        bool IsAbusiveLanguage(PostData postData);
    }
}



using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace AbusiveLanguageDetection.TextProcessing.Classifier.NaiveBayes
{
    public static class ClassifierHelper
    {
        public static readonly HashSet<string> ExcludeList = new
HashSet<string>(StringComparer.OrdinalIgnoreCase);
        public static readonly Dictionary<char, char> TokenCharMappings;

        static ClassifierHelper()
        {
            //Tokens to exclude
            string excludedTokens = Configuration.ExcludedTokensList;

            if (!string.IsNullOrWhiteSpace(excludedTokens))
            {
                excludedTokens = Path.Combine(Configuration.BaseFolder,
excludedTokens);
                if (!File.Exists(excludedTokens))
                    throw new FileNotFoundException("Unable to find
Excluded Files List at " + excludedTokens);

                using (StreamReader file = new
StreamReader(excludedTokens))
                {
                    string line;
                    while ((line = file.ReadLine()) != null)
                    {
                        if (string.IsNullOrWhiteSpace(line.Trim()))
continue;

                        if (line.StartsWith("--")) continue;
```

```
                        if (!ExcludeList.Contains(line.Trim()))
                            ExcludeList.Add(line);
                    }
                }
            }

            char[] alphabets = "abcdefghijklmnopqrstuvwxyz".ToCharArray();
            char[] numerics = "0123456789".ToCharArray();

            TokenCharMappings = alphabets.ToDictionary(alpha => alpha);
            foreach (char number in numerics)
            {
                TokenCharMappings.Add(number, ' ');
            }

        }
    }
}


using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;

namespace AbusiveLanguageDetection.TextProcessing.Classifier.NaiveBayes
{
    public class Evidence : IEvidence
    {
        private const char EvidenceSeparator = ',';

        private readonly Dictionary<string, double> _evidences;
        private readonly string _evidenceFileName;

        private long _totalWords;
        private readonly string _className;
        private readonly bool _saveEvidence;

        public string Name
        {
            get { return _className; }
        }
        public long TotalWords
        {
            get { return _totalWords; }
        }

        public Evidence(string className, string evidenceFilePath, bool
loadEvidence = true, bool saveEvidence = false)
        {
            if (string.IsNullOrWhiteSpace(className)) throw new
Exception("Class name was not defined");

            _saveEvidence = saveEvidence;
            _className = className;
            _evidences = new Dictionary<string,
double>(StringComparer.OrdinalIgnoreCase);
            _evidenceFileName = evidenceFilePath;
            if (loadEvidence) LoadEvidenceFromCache(evidenceFilePath);
        }

        ~Evidence()
        {
```

```
            if (_saveEvidence) PersistEvidence();
        }

        public void LoadEvidenceFromCache(string filePath)
        {
            if (!File.Exists(filePath)) return;


            using (StreamReader file = new StreamReader(filePath))
            {
                string line;
                while ((line = file.ReadLine()) != null)
                {
                    string[] keyValue = line.Split(EvidenceSeparator);
                    if (!_evidences.ContainsKey(keyValue[0]))
                    {
                        long value = long.Parse(keyValue[1]);
                        _evidences.Add(keyValue[0], value);
                        _totalWords += value;
                    }
                    else
                    {
                        throw new Exception(
                            $"Duplicate entries of {keyValue[0]} found
while loading evidences from {filePath}");
                    }
                }
            }
        }

        public bool AddEvidenceData(string trainingData, HashSet<string>
wordsToIgnore)
        {
            if (string.IsNullOrWhiteSpace(trainingData)) return false;

            List<string> tokens =
Tokenizer.TokenizeNow(trainingData).ToList();

            foreach (
                string token in
                tokens.Where(token =>  !string.IsNullOrWhiteSpace(token) &&
!wordsToIgnore.Contains(token)))
            {
                if (!_evidences.ContainsKey(token))
                    _evidences[token] = 0;

                _evidences[token]++;
                _totalWords++;
            }

            return true;
        }

        public bool AddEvidenceData(IEnumerable<string> trainingData,
HashSet<string> wordsToIgnore)
        {
            foreach (string data in trainingData)
            {
                AddEvidenceData(data, wordsToIgnore);
            }

            return true;
        }
```

```
        public IDictionary<string, double> GetEvidence()
        {
            return _evidences;
        }

        public bool PersistEvidence(bool backupExisting = true)
        {
            if (_evidences == null || _evidences.Count <= 0)
                return false;

            if (backupExisting)
                BackupFile(_evidenceFileName);

            using (StreamWriter file = new StreamWriter(_evidenceFileName))
            {
                foreach (
                    string line in
                    _evidences.Select(
                        evidence =>
                            evidence.Key.Trim() + "," +
evidence.Value.ToString(CultureInfo.InvariantCulture).Trim())
                )
                {
                    file.WriteLine(line);
                }
            }

            return true;
        }

        public static void BackupFile(string fullFileName)
        {
            if (!File.Exists(fullFileName))
            {
                return;
            }

            string ext = Path.GetExtension(fullFileName);
            string newFileName = fullFileName +
DateTime.Now.ToString(".yyyy.MM.dd.HH.mm.ss.fff") + ext;
            File.Copy(fullFileName, newFileName);
        }
    }

}


using System;
using System.Collections.Generic;
using System.Diagnostics;

namespace AbusiveLanguageDetection.TextProcessing.Classifier.NaiveBayes
{
    public static class Extensions
    {
        public static IEnumerable<IEnumerable<T>> Chunk<T>(this
IEnumerable<T> source,
                                                           int chunkSize)
        {
            if (source == null)
                throw new ArgumentNullException(nameof(source));
            if (chunkSize <= 0)
                throw new ArgumentOutOfRangeException(nameof(chunkSize),
```

```
                                                    "The chunkSize
parameter must be a positive value");

            return source.ChunkInternal(chunkSize);
        }

        private static IEnumerable<IEnumerable<T>> ChunkInternal<T>(
            this IEnumerable<T> source, int chunkSize)
        {
            Debug.Assert(source != null);
            Debug.Assert(chunkSize > 0);

            using (IEnumerator<T> enumerator = source.GetEnumerator())
                do
                {
                    if (!enumerator.MoveNext())
                        yield break;

                    yield return ChunkSequence(enumerator, chunkSize);
                } while (true);
        }

        private static IEnumerable<T> ChunkSequence<T>(IEnumerator<T>
enumerator,
                                                       int chunkSize)
        {
            Debug.Assert(enumerator != null);
            Debug.Assert(chunkSize > 0);

            int count = 0;

            do
            {
                yield return enumerator.Current;
            } while (++count < chunkSize && enumerator.MoveNext());
        }
    }
}


using System.Collections.Generic;

namespace AbusiveLanguageDetection.TextProcessing.Classifier.NaiveBayes
{
    public interface IEvidence
    {
        string Name { get; }
        long TotalWords { get; }
        bool AddEvidenceData(string trainingData, HashSet<string>
wordsToIgnore);
        bool AddEvidenceData(IEnumerable<string> trainingData,
HashSet<string> wordsToIgnore);
        IDictionary<string, double> GetEvidence();
        bool PersistEvidence(bool backupExisting);
        void LoadEvidenceFromCache(string filePath);
    }
}


using System;
using System.Collections.Generic;
using System.Linq;

namespace AbusiveLanguageDetection.TextProcessing.Classifier.NaiveBayes
```

```
{
    public class NaiveBayesClassifier : IClassifier
    {
        private readonly IEvidence _classA;
        private readonly IEvidence _classB;

        private const int WordsInChunk = 50;

        public IEvidence ClassA
        {
            get { return _classA; }
        }

        public IEvidence ClassB
        {
            get { return _classB; }
        }

        public NaiveBayesClassifier(IEvidence first, IEvidence second)
        {
            _classA = first;
            _classB = second;
        }

        public IDictionary<string, double> Classify(List<string> content,
HashSet<string> wordsToIgnore)
        {
            if (_classA == null || _classB == null)
                throw new Exception("One of the evidences were not properly
defined");
            if (string.IsNullOrWhiteSpace(_classA.Name))
                throw new Exception("Evidence name not defined on the first
one");
            if (string.IsNullOrWhiteSpace(_classB.Name))
                throw new Exception("Evidence name not defined on the
second one");

            double chunkSize = Math.Ceiling(content.Count /
(double)WordsInChunk);

            var scores = new List<Dictionary<string, double>>();
            for (int i = 0; i < chunkSize; i++)
            {
                var score = new Dictionary<string, double>
                {
                    {_classA.Name, 0.0},
                    {_classB.Name, 0.0}
                };

                scores.Add(score);
            }
            int index = 0;
            foreach (IEnumerable<string> wordsChunk in
content.Chunk(WordsInChunk))
            {
                foreach (
                    string word in
                        wordsChunk.Where(word =>
!string.IsNullOrWhiteSpace(word) && !wordsToIgnore.Contains(word)))
                {
                    //First Class
                    IDictionary<string, double> classAEvidence =
_classA.GetEvidence();
                    double wordCountInClassA;
```

```
                    wordCountInClassA = classAEvidence.TryGetValue(word,
out wordCountInClassA)
                        ? wordCountInClassA
                        : 0.01; //ToDo - Make this 1 or 0.01

                    double scoreClassA = Math.Log(wordCountInClassA /
_classA.TotalWords);
                    scores[index][_classA.Name] += scoreClassA;

                    //Second Class
                    IDictionary<string, double> classBEvidence =
_classB.GetEvidence();
                    double wordCountInClassB;
                    wordCountInClassB = classBEvidence.TryGetValue(word,
out wordCountInClassB)
                        ? wordCountInClassB
                        : 0.01; //ToDo - Make this 1 or 0.01
                    double scoreClassB = Math.Log(wordCountInClassB /
_classB.TotalWords);
                    scores[index][_classB.Name] += scoreClassB;
                }

                long totalWordsAllCategories = _classA.TotalWords +
_classB.TotalWords;

                scores[index][_classA.Name] +=
Math.Log((double)_classA.TotalWords / totalWordsAllCategories);
                scores[index][_classB.Name] +=
Math.Log((double)_classB.TotalWords / totalWordsAllCategories);

                double scoreA = Math.Exp(scores[index][_classA.Name]);
                double scoreB = Math.Exp(scores[index][_classB.Name]);
                double totalScore = scoreA + scoreB;

                scores[index][_classA.Name] = scoreA / totalScore;
                scores[index][_classB.Name] = scoreB / totalScore;

                index++;
            }

            var results = new Dictionary<string, double>
                            {
                                {_classA.Name, 0.0},
                                {_classB.Name, 0.0}
                            };

            foreach (Dictionary<string, double> score in scores)
            {
                results[_classA.Name] += score[_classA.Name];
                results[_classB.Name] += score[_classB.Name];
            }

            results[_classA.Name] = results[_classA.Name] / scores.Count;
            results[_classB.Name] = results[_classB.Name] / scores.Count;

            return results;
        }

    }
}



using System.Collections.Generic;
```

```
namespace AbusiveLanguageDetection.TextProcessing.Classifier
{
    public interface IClassifier
    {
        IDictionary<string, double> Classify(List<string> content,
HashSet<string> wordsToIgnore);
    }
}


using System.Collections.Generic;
using System.Text;

namespace AbusiveLanguageDetection.TextProcessing.Emoji
{
    public class EmojiHandler : IEmojiHandler
    {
        private readonly Dictionary<string, string> _emojiDictionary;

        public EmojiHandler()
        {
            _emojiDictionary = new Dictionary<string, string>
            {


            };
        }
    }
}

namespace AbusiveLanguageDetection.TextProcessing.Emoji
{
    public interface IEmojiHandler
    {
        string HandleEmoji(string content);
    }
}


namespace AbusiveLanguageDetection
{
    public static class Configuration
    {
        public static string BaseFolder =

@"C:\Users\Anna\Downloads\AbusiveLanguageDetection\AbusiveLanguageDetection
\Repo";
        public static string ExcludedTokensList = "ExcludedTokens.txt";
        public static string Sentenaizer = "en-sent.bin";
        public static string Tokeneizer = "en-token.bin";
        public static string SpecialSymbols = "SpecialSymbols.csv";

        public static string NeutralClass = "Neutral";
        public static string AbusiveClass = "Abusive";

        public static string LabeledData = "LabeledData.csv";

        public static bool UseEmojiHandler = false;
        public static bool UseSpecialSymbolHandler = true;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using AbusiveLanguageDetection.ConsoleTest.LabeledDatas;
using AbusiveLanguageDetection.TextProcessing.Classifier.NaiveBayes;

namespace AbusiveLanguageDetection.ConsoleTest.Teaching
{
    public static class Teacher
    {
        public static Tuple<IEvidence, IEvidence>
TeachClassifier(List<LabeledData> inputData)
        {
            string neutralEvidencePath =
Path.Combine(Configuration.BaseFolder, Configuration.NeutralClass);
            IEvidence neutralEvidence = new
Evidence(Configuration.NeutralClass, neutralEvidencePath, false, true);

            string abusiveEvidencePath =
Path.Combine(Configuration.BaseFolder, Configuration.AbusiveClass);
            IEvidence abusiveEvidence = new
Evidence(Configuration.AbusiveClass, abusiveEvidencePath, false, true);

            neutralEvidence.AddEvidenceData(inputData.Where(x =>
!x.IsOffensive).Select(x => x.Text), ClassifierHelper.ExcludeList);
            abusiveEvidence.AddEvidenceData(inputData.Where(x =>
x.IsOffensive).Select(x => x.Text), ClassifierHelper.ExcludeList);

            return new Tuple<IEvidence, IEvidence>(neutralEvidence,
abusiveEvidence);
        }
    }
}


namespace AbusiveLanguageDetection.ConsoleTest.LabeledDatas
{
    public class LabeledData
    {
        public readonly bool IsOffensive;
        public readonly string Text;

        public LabeledData(bool isOffensive, string text)
        {
            IsOffensive = isOffensive;
            Text = text;
        }
    }
}


using System;
using System.Collections.Generic;
using System.IO;
using CsvHelper;

namespace AbusiveLanguageDetection.ConsoleTest.LabeledDatas
{
    public static class LabeledDataReader
    {
        public static List<LabeledData> ParseLabeledData(string fileName)
        {
            List<LabeledData> result = new List<LabeledData>();
```

```
            using (var txtReader = new StreamReader(fileName))
            {
                bool wasHeaderRead = false;
                var csvHelper = new CsvReader(txtReader);
                while (csvHelper.Read())
                {
                    if (csvHelper.Context.Record.Length != 7)
                    {
                        throw new ArgumentException("Expected 6 fields in
csv");
                    }
                    if (!wasHeaderRead)
                    {
                        wasHeaderRead = true;
                        continue;
                    }
                    var txtClass =
csvHelper.GetField<LabeledDataClasses>(5);
                    string txt = csvHelper.GetField<string>(6);
                    var tmp = new LabeledData(txtClass !=
LabeledDataClasses.Neither, txt);
                    result.Add(tmp);
                }
            }
            return result;
        }
    }
}
```

Додаток 2

Копія презентації